Average-Case Intractable NP Problems
Jie Wang

Contents

Average-Case Intractable NP Problems

Jie Wang*
UNC Greensboro †

Abstract

The notion of NP-completeness has provided a rigorous mathematical definition for measuring intractability of NP problems. But this measure applies only to worst-case complexity. Being NP-complete does not indicate that a problem is intractable on the average case. Indeed, some NP-complete problems are "easy on average," though some may not be. Levin [Lev86] initiated the study of average-case NP-completeness to measure average-case intractability. He showed that a bounded tiling problem under a simple distribution is average-case NP-complete. Since then, several additional average-case NP-complete problems have been shown within Levin's framework. This paper is intended to provide a comprehensive survey of average-case NP-complete problems that have been published so far, and the techniques of obtaining these results.

1 Introduction

The notion of NP-completeness, introduced by Cook [Coo71] and independently by Levin [Lev73], has provided a rigorous mathematical definition for measuring intractability of NP problems. Karp [Kar72] introduced the methodology of many–one reductions and demonstrated the rich variety of NP-complete problems; and just before 1979, several hundreds of additional NP-complete problems were discovered [GJ79]. Since then, many more computational problems from almost all areas involve computing have been shown to be NP-complete.

Despite many years of intensive study, no efficient algorithms have been found for NP-complete problems, and so NP-complete problems are generally thought of as being computationally intractable. However, NP-completeness is a worst-case concept that does not provide information about how difficult an NP-complete problem might be on the average case. Indeed, several NP-complete problems have been shown to be tractable "on average." For example, although HAMILTONIAN PATH is NP-complete, Gurevich and Shelah [GS87] have shown that if a graph is chosen in a certain reasonable way, then the HAMILTONIAN PATH problem can be solved by a deterministic algorithm

*Supported in part by NSF under grant CCR-9424164. Support was also provided by the University of North Carolina at Greensboro in the form of research leave.

†Department of Mathematical Sciences, University of North Carolina at Greensboro, Greensboro, NC 27412, USA. E-mail: wang@uncg.edu.

whose expected running time on graphs with n vertices is bounded by a linear polynomial in n. Also, despite the fact that GRAPH 3-COLORABILITY is NP-complete, Wilf [Wil84] has shown that it can be solved by a deterministic algorithm whose expected running time on graphs with n vertices under a uniform distribution is actually bounded by a constant. Thus, the average-case complexity of a problem is, in many respects, a more significant measure than its worst-case complexity.

Given a problem and a distribution on instances, finding an expected polynomial-time algorithm to solve the problem or proving that such an algorithm does not exist is an important issue. There are two central notions needed in studying this issue along similar lines to the theory of NP-completeness. Namely, a notion for measuring efficiency on the average case and a notion of completeness for measuring intractability on the average case.

Expected polynomial time is a natural notion to measure average efficiency of an algorithm. But it is, among other things, machine dependent (see, for example, [Gur91, Ven91, Wan96]), and so it cannot be used to build a general theory on average-case intractability for NP problems. To overcome this obstacle, Levin defined a robust notion on what it means for the running time of a deterministic algorithm to be polynomial on average with respect to the input distribution. A problem with an associated probability distribution is called a *distributional problem*. A distributional problem (A, μ) is said to be in AP if A can be solved by a deterministic algorithm whose running time is polynomial on average with respect to μ. Levin then defined reductions between distributional problems in such a way that reductions are transitive and if a distributional problem is reducible to a second distributional problem which is in AP, then the original distributional problem is also in AP. With this machinery in place, Levin showed that distributional TILING with a simple distribution is average-case NP-complete, meaning that it is not in AP unless every NP problem under every polynomial-time computable distribution is in AP. Since then, several additional average-case NP-complete problems have been found. Among them are the distributional Post correspondence problem [Gur91] and the distributional word problem for finitely presented groups [Wan95b].

However, as pointed out by Gurevich [Gur87, Gur91], the reduction defined in [Lev86] has certain limitations. Gurevich showed that using such a reduction, a distributional problem under a "flat" distribution cannot be complete unless nondeterministic exponential time collapses to deterministic exponential time. A distribution μ is *flat* if there exists an $\epsilon > 0$ such that for all x, $\mu(x) \leq 2^{-|x|^\epsilon}$, where $|x|$ denotes the size of x. Wang and Belanger [WB95] further showed that, without any assumption, a distributional problem under a flat distribution cannot be complete under one-one and p-length-preserving reductions.[1] To overcome this obstacle, Venkatesan and Levin [VL88] introduced the notion of randomized reductions, and they showed that under such a reduction, a graph edge coloring problem with a flat distribution is average-case

[1] A reduction f is called *p-length-preserving* if it preserves the size of the output within a polynomial of the size of the input. Namely, there is a polynomial p such that for all x, $p(|f(x)|) \geq |x|$. Such a length-preserving reduction is also called *p-honest* in the literature.

NP-complete. Several additional average-case NP-complete problems with flat distributions have been found since then. Among them are the distributional matrix transformation problem [Gur90, BG95] and the distributional matrix representability problem [VR92].

Distributions on instances of all these average-case NP-complete problems are simple in that the components of an instance are selected uniformly at random. Such distributions are polynomial-time computable. One might suspect that a more complex distribution would make it easier to generate hard instances of an NP problem and so additional average-case NP-complete problems could be found. This motivated Ben-David *et. al.* [BCGL92] to study p-samplable distributions. A distribution μ is *p-samplable* if there is a randomized algorithm that takes no inputs and outputs x with probability $\mu(x)$ in polynomial time of $|x|$. They then showed that for any standard NP-complete problem, there is a p-samplable distribution that makes it to be average-case complete. But this p-samplable distribution, constructed by an enumeration technique, is far from being considered as natural. Impagliazzo and Levin [IL90] later showed that for NP search problems, p-samplable distributions do not generate harder instances than simply picking instances uniformly at random. In particular, they showed that any NP search problem with a p-samplable distribution is reducible to an NP search problem with a uniform distribution under a randomized reduction. For decision problems, the same result can be obtained using a randomized truth-table reduction [BCGL92] (see also [Wan96]). Hence, without loss of generality, we shall only focus on polynomial-time computable distributions.

Two other average time measurements have been recently proposed in [RS93] and [CS96], respectively, in an effort to refine Levin's notion of average time. The definition of average time proposed in [RS93] does not provide harder NP problems [BW95, BW]. The definition of average time proposed in [CS96] provides the same AP as Levin's under a certain reasonable condition on distributions [CS96], and that condition is satisfied in the discussion here. Thus, as far as average-case NP-completeness is concerned, Levin's average time measurement is sufficient, which is also easier to work with.

This paper is organized as follows. Basic definitions and results are presented in Section 2. A list of average-case NP-complete problems are presented in Section 3. Completeness proofs of these problems are given in Sections 4, 5, and 6. Some final remarks and open problems are given in Section 7.

2 Preliminaries

The reader is referred to [Wan96] for a comprehensive survey of the mathematical theory of average-case computational complexity. For convenience, we provide below some basic definitions and results.

We use the binary alphabet $\Sigma = \{0, 1\}$ for encoding strings. Denote by $|x|$ the length of a binary string x. Denote by $\leq$ the standard lexicographical order on Σ^*. A *probability distribution* μ is a real-valued function from Σ^* to [0,1] such that $\sum_x \mu(x) = 1$. We assume that μ on the empty string is always 0. We may also use *distribution*, *probability*, *weight*, or *density* to denote

probability distribution. The *distribution function* of μ, denoted by μ^*, is defined by $\mu^*(x) = \sum_{y \leq x} \mu(y)$, where $\leq$ is the standard lexicographical order on Σ^*. For a function f, we use $f^\varepsilon(x)$ to denote $(f(x))^\varepsilon$ for $\varepsilon > 0$ and use f^{-1} to denote the inverse of f. We use the standard notations O, Θ, and Ω (see, for example, [CLR90]).

2.1 Average Polynomial Time

Levin [Lev86] started with the following definition to measure average time efficiency.

Definition 2.1 A function $f : \Sigma^+ \to \mathbb{N}$ is *polynomial on μ-average* if there exists an $\varepsilon > 0$ such that $\sum_x f^\varepsilon(x)|x|^{-1}\mu(x) < \infty$.

This definition is robust and machine independent. Also, if a function is an expected polynomial over distribution μ, then it is polynomial on μ-average. The reader is referred to [Gur89, Gur91, Ven91, Wan96] for motivation and justification of this definition.

A problem is solvable in average polynomial time with respect to distribution μ if it can be solved by a *deterministic* algorithm whose running time is polynomial on μ-average. A problem with an associated probability distribution is called a *distributional problem*.

2.2 Polynomial-Time Reductions

Given two distributional problems, we wish to know which one is computationally more difficult. The standard technique for such comparisons is to find a reduction from one problem to another. For simplicity, we will focus on distributional decision problems. Such a problem is comprised of a set of instances which are either positive or negative for the underlying decision problem, and a probability distribution on these instances. It is customary to only specify the set of positive instances, and we are only concerned with instances with positive probability. Under such a convention, we use (D, μ) to denote a distributional decision problem, where D is the set of all positive instances x with $\mu(x) > 0$. The problem is to decide, for a given instance x with $\mu(x) > 0$, whether $x \in D$. If $D \in$ NP, then (D, μ) is called a distributional NP decision problem.

Denote by AP the class of all distributional decision problems (D, μ) such that D is solvable in polynomial time on μ-average.

The following definitions are due to Levin [Lev86] and their current forms are from [Gur91].

Definition 2.2 Let μ and ν be distributions over the same domain. Then μ is *dominated by* ν, written as $\mu \preceq \nu$, if there exists a polynomial p such that for all x, $\mu(x) \leq p(|x|)\nu(x)$.

Definition 2.3 Let μ and ν be distributions on the instances of the decision problems A and B, respectively, and f be a reduction from A to B. Then μ is *dominated by ν with respect to f*, written as $\mu \preceq_f \nu$, if there exists a distribution

μ_1 on A such that $\mu \preceq \mu_1$ and $\nu(y) = f(\mu_1)(y)$ for all $y \in \text{range}(f)$, where $f(\mu_1)(y) = \sum_{f(x)=y} \mu_1(x)$.

Definition 2.4 (A, μ) is *polynomial-time reducible* to (B, ν) if there is a polynomial-time computable reduction f such that A is many–one reducible to B via f and $\mu \preceq_f \nu$.

The following lemma is straightforward [Gur91].

Lemma 2.1 *If a reduction f is one-one, then $\mu \preceq_f \nu$ iff $\mu \preceq \nu \circ f$.*

In the sequel, we will often use "p-time" to denote "polynomial-time", and "ap-time" to denote "average polynomial-time."

The following lemma is straightforward (see, for example, [Gur91, Wan96]).

Lemma 2.2 *(1) If (A, μ) is p-time reducible to (B, ν) and (B, ν) is in* AP*, then so is (A, μ). (2) The p-time reductions are transitive.*

As indicated in [Lev86] and discussed in [Gur91], instead of requiring a reduction f be p-time computable, one only needs to require that f be computable in ap-time. This weakens the original definition in two ways: the reduction is weaker and the domination is weaker. The distribution μ is *weakly dominated by* the distribution ν if there is a function g such that for all x, $\mu(x) \leq g(x)\nu(x)$, where g is polynomial on μ-average.

Definition 2.5 (A, μ) is *ap-time reducible* to (B, ν) if there is a many–one reduction f that reduces A to B such that f is computable in time polynomial on μ-average and μ is weakly dominated by μ_1 for some μ_1 such that for all x, $\nu(f(x)) = f(\mu_1)(f(x))$.

Lemma 2.3 *(1) If (A, μ) is ap-time reducible to (B, ν) and (B, ν) is in* AP*, then so is (A, μ). (2) The ap-time reductions are transitive.*

2.3 Polynomial-Time Computable Distributions

Let (D, μ) be a distributional NP problem. If every other distributional NP problem is p-time reducible to it, then (D, μ) has no ap-time algorithm unless every NP problem under any allowable distribution has one. In order for such a complete problem to exist, a certain condition on the computability of allowable distributions is needed. If arbitrary distributions are allowed, Wang and Belanger [WB93a] showed that no complete distributional problems can exist with respect to one-one, p-time reductions. Note that all the known NP-complete problems are complete via one-one, p-time reductions.

Levin [Lev86] suggested that it is reasonable to require distributions be p-time computable. A real-valued function $f\colon \Sigma^* \to [0, 1]$ is p-time computable [Ko83] if there is a deterministic algorithm which, on every string x and every positive integer k, outputs a finite binary fraction y in time bounded by a polynomial in $|x|$ and k and such that $|f(x) - y| \leq 2^{-k}$. Clearly, if μ^* is p-time computable then so is μ. Blass showed that the converse is not true unless

P = NP (see [Gur91]). With this fact in mind, we assume that in the sequel, when we say that a distribution μ is p-time computable we mean that both μ and μ^* are p-time computable.

There is strong evidence to believe, as hypothesized by Levin (see [Joh84]), that any "natural" probability distribution is either p-time computable or else is dominated by one that is. One can indeed verify that all commonly used distributions do satisfy this property.

Denote by DistNP the class of all distributional NP problems (D, μ) such that $\mu \preceq \nu$ for some p-time computable distribution ν. (Other names such as RNP [Gur91] and DNP [WB93a] have also been used to denote DistNP.)

2.4 Uniform Distributions

Distributions may be defined on all binary strings by first selecting lengths and then selecting strings of that length. Although it is mathematically impossible to select strings with equal chance from an infinite sample space, strings of the same length can be selected with equal likelihood. It is also impossible to select integers from $\mathbb{N}$ with the same probability, but one can select an integer with a probability close to being "uniform." A p-time computable distribution μ on Σ^* is called *uniform* if for all x, $\mu(x) = \rho(|x|)2^{-|x|}$, where $\sum_n \rho(n) = 1$ and there is a polynomial p such that for all but finitely many n, $\rho(n) \geq 1/p(n)$.

It is important to note that for the purpose of proving completeness results, $\rho(n) \geq 1/p(n)$ is the only requirement needed since domination allows a polynomial factor, and so some longer strings can certainly be given more weights than shorter ones. Levin [Lev86] used n^{-2} for $\rho(n)$ for notational convenience (normalized by dividing by $\sum_n n^{-2} = \pi^2/6$), and $|x|^{-2}2^{-|x|}$ is often referred to as the default uniform distribution.

2.5 Distribution Controlling Lemma

Distributions that are p-time computable can be dominated (with respect to p-time computable functions) by uniform distributions, which is an important property for proving completeness results. Levin [Lev86] first proved this property using a "perfect rounding" technique. Gurevich [Gur91] provided a different and easier proof, based on that Wang and Belanger [WB95] further showed that if the distribution is not too small, it will also dominate the same uniform distribution within a constant factor.

Lemma 2.4 (Distribution Controlling Lemma) *Let μ be a p-time computable distribution.*

1. *There exists a total, one-one, p-time computable and p-time invertible function $\alpha\colon \Sigma^* \to \Sigma^*$ such that for all x, $\mu(x) < 4 \cdot 2^{-|\alpha(x)|}$.*
2. *If there exists a polynomial p such that for all x, $\mu(x) > 2^{-p(|x|)}$, then there is a total, one-one, p-time computable and p-time invertible function $\beta\colon \Sigma^* \to \Sigma^*$ such that for all x, $4 \cdot 2^{-|\beta(x)|} \leq \mu(x) < 20 \cdot 2^{-|\beta(x)|}$.*

3 Average-Case NP-Complete Problems

Given below is a list of best-known average-case NP-complete problems that have been published so far.

3.1 Distributional Halting Problem (Version 1)

Let $M_1, M_2, \ldots$ be a fixed enumeration of nondeterministic Turing machines in which the index i is an integer that codes up the symbols, states, and transition table of the i-th Turing machine M_i.

DISTRIBUTIONAL HALTING PROBLEM (VERSION 1)

Instance. Binary strings i, x, and a positive integer n, where i is a positive integer in binary form. The size of the instance is $|i| + |x| + n$.

Question. Does M_i accept x within n steps?

Distribution. Proportional to $2^{-(l+m)}l^{-2}m^{-2}n^{-2}$, where $l = |i|$ and $m = |x|$. This corresponds to the following experiment: Randomly and independently choose binary strings i, x, and a positive integer n with respect to the default uniform distributions. [2]

The distributional halting problem was shown by Gurevich [Gur91] to be average-case NP-complete. The version presented here is from [BCGL92, WB93a].

3.2 Distributional Tiling Problem

A tile is a square with a symbol on each side which may not be rotated or turned over. We assume that there are infinite copies of each tile. By a tiling of an $n \times n$ square it means an arrangement of n^2 tiles covering the square in which the symbols on the common sides of adjacent tiles are the same. The size of a tile is the length of the binary representation of the four symbols at each edge in a fixed order.

DISTRIBUTIONAL TILING PROBLEM

Instance. A finite set of tiles T, a positive integer n, and a sequence $S = s_1 s_2 \ldots s_k$ ($k \leq n$) of tiles that match each other, *i.e.* the symbol on the right side of s_i is the same as that on the left side of s_{i+1}. The size of the instance is n plus the sum of the sizes of all members in T plus the sum of the sizes of all members in S.

Question. Can S be extended to tile an $n \times n$ square using tiles from T?

Distribution. Proportional to $\mathbf{Pr}[T]n^{-2}\mathbf{Pr}[S]$, where $\mathbf{Pr}[T]$ is the probability of T (one can use one's favorite distribution to select T or simply select one uniformly at random among binary strings since T is coded in binary) and $\mathbf{Pr}[S]$ is the probability of choosing S, where S is chosen by choosing k at

[2] Note that the index i here is selected uniformly as a binary string. How to select i, however, is not important to obtain the completeness result. One can use one's favorite distributions to select it, for example, one may select states and transition functions under different distributions.

random with probability $1/n$, choosing the first tile s_1 at random from T, and choosing the s_i $(i > 1)$ sequentially and uniformly at random from those tiles in T that match s_{i-1}.

Levin [Lev86] originally considered the distributional tiling problem for tiles with marked corners, namely, the corners of tiles, instead of edges, are marked with letters. In the tiling of an $n \times n$ square, letters on the touching corners of adjacent tiles are the same.

The distributional tiling problem for tiles with marked corners was shown to be average-case NP-complete by Levin [Lev86], but only a proof sketch was given. A detailed proof for tiles with marked edges was given in [Gur91] (see also [BW93, WB95]).

3.3 Distributional Post Correspondence and Related Problems

The Post correspondence problem was first studied by Post (see, for example, [Dav77]), which is to decide, when given a nonempty list $L = \langle (u_1, v_1), ..., (u_m, v_m) \rangle$ of pairs of binary strings, whether there is a sequence of integers $i_1, i_2, \ldots, i_k$, with $k \geq 1$, such that $u_{i_1} u_{i_2} \cdots u_{i_k} = v_{i_1} v_{i_2} \cdots v_{i_k}$. The sequence $i_1, i_2, \ldots, i_k$ is called a solution of length k to this instance.

Distributional Post Correspondence Problem

Instance. A nonempty list $L = \langle (u_1, v_1), ..., (u_m, v_m) \rangle$ of pairs of binary strings, and a positive integer n. The size of the instance is $n + \sum_{i=1}^{m} (|u_i| + |v_i|)$.

Question. Is there a solution to L of length at most n?

Distribution. Randomly and independently choose positive integers n and m, then randomly and independently choose binary strings $u_1, v_1, ..., u_m, v_m$. The random choices are made with respect to the default uniform probability distributions on positive integers and binary strings. Hence, the probability distribution $\mu_C(L(m), n)$ is proportional to

$$\left(\prod_{i=1}^{m} \frac{2^{-(|u_i|+|v_i|)}}{|u_i|^2 |v_i|^2} \right) \left(\frac{1}{m^2} \right) \left(\frac{1}{n^2} \right).$$

The distributional Post correspondence problem was shown to be average-case NP-complete by Gurevich [Gur91].

By a slight modification of the distributional Post correspondence problem, Gurevich [Gur90] (see also [BG95]) defined the following correspondence problem for strings and showed that it is average-case NP-complete.

Distributional String Correspondence Problem

Instance. A binary string a, a nonempty list $L(m) = \langle (u_1, v_1), ..., (u_m, v_m) \rangle$ of binary string pairs, and a positive integer n. The size of the instance is $|a| + n + \sum_{i=1}^{m} (|u_i| + |v_i|)$.

Question. Are there strings $x = u_{i_1} u_{i_2} \cdots u_{i_k}$ and $y = v_{i_1} v_{i_2} \cdots v_{i_k}$, $k \leq n$, such that $ax = y$?

Distribution. Proportional to $|a|^{-2}2^{-|a|}\mu_C(L(m), n)$.

DISTRIBUTIONAL PALINDROME PROBLEM

Instance. A context-free grammar with productions

$$T \to u_1 T v_1 \mid \cdots \mid u_m T v_m \mid e,$$

and a positive integer n, where u_i and v_i are binary strings, and e is the empty string. The size of the instance is $n + \sum_{i=1}^{m}(|u_i| + |v_i|)$.

Question. Can a nonempty palindrome string be derived within n steps?

Distribution. Randomly and independently choose positive integers m and n, then randomly and independently choose binary strings $u_1, v_1, \ldots, u_m, v_m$ with respect to the default uniform distributions. Let $L(m) = \{(u_1, v_1), \ldots, (u_m, v_m)\}$, then the probability distribution of an palindrome instance is the same as $\mu_C(L(m), n)$.

The distributional palindrome problem was shown to be average-case NP-complete by Gurevich [Gur91].

3.4 Distributional String-Rewriting Problems

We first consider the word problem for Thue systems. This problem was first studied in modern algebra by Thue [Thu14]. Wang and Belanger [WB95] studied its bounded version with uniform distributions on instances. Let A be a non-empty finite alphabet. An ordered pair $\langle g, h\rangle$ of strings on alphabet A is called a production (rule, or rewriting rule). It is customary to write the production in the form $g \to h$. Let u and v be two strings. Let $g \to h$ be a production. Write $u \Rightarrow_{g\to h} v$ (omitting $g \to h$ when there is no ambiguity) if $u = agb$ and $v = ahb$ for (possible null) strings a and b. Let Π be a set of productions. Then write $u \Rightarrow_\Pi v$ if $u \Rightarrow_p v$ for some $p \in \Pi$. Write $u \Rightarrow_\Pi^n v$ if there is a finite sequence: $u = u_1 \Rightarrow_\Pi u_2 \Rightarrow_\Pi \cdots \Rightarrow_\Pi u_k = v$ for $k \leq n$. The inverse of the production $g \to h$ is the production $h \to g$. A Thue system Π is a non-empty set of productions such that for each $P \in \Pi$, the inverse of P is also in Π. Write (g, h) to denote both the production $g \to h$ and its inverse. Let Π be a Thue system. Let u and v be two strings. Then $u \Rightarrow_\Pi v$ iff $v \Rightarrow_\Pi u$. So we write $u =_\Pi v$ if $u \Rightarrow_\Pi v$. We write $u =_\Pi^n v$ if there is a finite sequence: $u = u_1 =_\Pi u_2 =_\Pi \cdots =_\Pi u_k = v$ for $k \leq n$. We write $u =_\Pi^* v$ if there exists an n such that $u =_\Pi^n v$. This is an equivalence relation, and set of equivalence classes form a monoid under the operation of concatenation.

Let $S = \{s_1, \ldots, s_m\}$ be a finite set of strings. We use $\| S \|_0$ to denote $\sum_{i=1}^{m} |s_i|$ and $\| S \|_1$ to denote $\prod_{i=1}^{m} |s_i|$.

DISTRIBUTIONAL WORD PROBLEM FOR THUE SYSTEMS

Instance. Thue system $\Pi = \{(g_1, h_1), ..., (g_m, h_m)\}$ of m productions, two binary strings u, v, and a positive integer n, where g's and h's are binary strings. The size of the instance is $n + |u| + |v| + \sum_{i=1}^{m}(|g_i| + |h_i|)$.

Question. Is $u =_\Pi^n v$?

Distribution. Randomly and independently choose positive integers n, m, and binary strings u, v. Then randomly and independently choose binary strings $g_1, h_1, ..., g_m, h_m$. The random choices are made with respect to the default uniform probability distributions on positive integers and binary strings. Hence, the probability distribution $\mu_W(\Pi, u, v, n)$ is proportional to

$$\frac{2^{-(\|\Pi\|_0+|u|+|v|)}}{(\|\Pi\|_1\ |u||v|n)^2}.$$

The distributional word problem for Thue systems was shown to be average-case NP-complete by Wang and Belanger [WB95].

Next, we consider other string rewriting problems. A set of productions is also called a *string-rewriting system*, in which the inverse of a production does not need to be in the system. We can similarly define the distributional word problem for semi-Thue systems on string-rewriting systems, which is average-case NP-complete [WB95]. The reader is referred to [BO93] for a comprehensive introduction to string-rewriting systems.

Distributional Common Ancestor Problem

Instance. A string-rewriting system R of m rewriting rules, two binary strings x and y, and a positive integer n. The size of the instance is $n + |x| + |y| + \sum_{(g,h)\in R}(|g| + |h|)$.

Question. Is there a binary string z such that $z \Rightarrow^n x$ and $z \Rightarrow^n y$?

Distribution. The same as $\mu_W(R, x, y, n)$.

Distributional Common Descendant Problem

Instance. A string-rewriting system R, two binary strings x and y, and a positive integer n. The size of the instance is $n + |x| + |y| + \sum_{(g,h)\in R}(|g| + |h|)$.

Question. Is there a binary string z such that $x \Rightarrow^n z$ and $y \Rightarrow^n z$?

Distribution. The same as $\mu_W(R, x, y, n)$.

These two string-rewriting problems were shown to be average-case NP-complete by Wang [Wan95b].

3.5 Distributional Word Problem for Groups

The word problem for groups was first considered by Dehn [Deh11] and Thue [Thu14], which is to decide, when given a group G and words x, y, whether x is equivalent to y in G. Novikov [Nov55] and Boone [Boo59] proved that there exists a finitely presented group with an unsolvable word problem.

A finite presentation of a group consists of a set of generators (abstract symbols) and a set of relations that relate the freely generated words. To be precise, let A be a finite set. The free group $[A]$ is a group including all elements that can be uniquely written as a reduced *word* in the form $a_1^{\pm}a_2^{\pm}\cdots a_n^{\pm}$, where $a_i \in A$, $a_i^{\pm}$ for a_i or a_i^{-1}, and no a_i appears adjacent to a_i^{-1}. When $n = 0$, we get the empty word e, which is the identity of the group. A word is positive if it does not contain negative components. The empty word e is regarded as a positive word. Positive words are also called strings. Words are multiplied by

juxtaposing with all expressions $a_i a_i^{-1}$ and $a_i^{-1} a_i$ canceled out until a reduced word is obtained. For each word w, the inverse word w^{-1} consists of all the symbols of w written in reverse order, where each a_i is replaced by a_i^{-1} and each a_i^{-1} is replaced by a_i. A relation on A is an expression $X = Y$, where X and Y are words on A. Let R be a finite set of relations on A. Let $\mathcal{K}_R$ denote the normal subgroup of $[A]$ generated by the XY^{-1} for $X = Y \in R$. A group G has a set of generators A and a set of relations R on A if G is isomorphic to the quotient group $[A]/\mathcal{K}_R$. A and R form a finite presentation of G, denoted by $[A; R]$. We extend the notation $[A; R]$ to allow several generators or sets of generators before the semicolon and several relations or sets of relations after the semicolon.

A quantifier may also be used to describe several similar relations in a compressed form. For instance, $\forall x \in A : x^3 = x^2$ represents relations $a_i^3 = a_i^2$ for $i = 1, 2, ..., n$. In this case, we say that $a_i^3 = a_i^2$ for a given i is a relation specified by the quantified statement $\forall x \in A : x^3 = x^2$. For simplicity, we use the term "quantified relation" for such a quantified statement.

Suppose $X = Y$ is a relation, then XY^{-1}, $X^{-1}Y$, YX^{-1}, and $Y^{-1}X$ are called relators. Obviously, if a is a generator, then aa^{-1} and $a^{-1}a$ are relators. Let u and v be (not necessarily reduced) words. Then $u \equiv v$ means that u and v have exactly the same spelling.

Following Tietze transformation theorem, Wang [Wan95a] defined the following elementary Tietze transformations for finitely presented group $[A; R]$, denoted by $\leftrightarrow_R$, where $\leftrightarrow_R$ is a symmetric operation. Let x be a word and y be a relator. (1) If x is an empty word, then $x \leftrightarrow_R y$. (2) If x is not empty and has spelling $x_1 x_2$ (x_1 or x_2 could be possibly null), then $x \leftrightarrow_R x_1 y x_2$.

Informally, this means that an equivalent word can be obtained by eliminating or introducing relators at any point of the original word. For simplicity, the following transformations is also considered to be elementary since such a transformation can be obtained by applying elementary transformations twice.

- If $x \equiv x_1 X x_2$, $x' \equiv x_1 X^{-1} x_2$ (x_1 and x_2 could be possibly null), and $X = Y$ is a relation, then $x \leftrightarrow_R x_1 Y x_2$ and $x' \leftrightarrow_R x_1 Y^{-1} x_2$.

The subscript R is often omitted from $\leftrightarrow_R$ when there is no confusion. Let $\overset{n}{\longleftrightarrow} = \leftrightarrow \circ \overset{n-1}{\longleftrightarrow}$. We use $\overset{*}{\longleftrightarrow}$ to denote $\overset{k}{\longleftrightarrow}$ for some k.

Let x and y be two words. x can be obtained from y in G in n steps if there is a sequence of n elementary Tietze transformations such that $x \overset{n}{\longleftrightarrow} y$ in G.

For a presentation $[A; R]$, we assume that words over A and relations (with or without quantifiers) in R are properly coded as binary strings.

Wang [Wan95a] studied the following distributional word problem for finitely presented groups, which is a slight modification of the original word problem of Dehn and Thue.

DISTRIBUTIONAL WORD PROBLEM FOR FINITELY PRESENTED GROUPS

Instance. A finitely presented group $G = [A; R]$, binary strings u, v, w, and a positive integer n, where $A = \{a_1, ..., a_l\}$ consists of generators and $R = \{r_1, ..., r_m\}$ consists of relations (with or without quantifiers), all coded in binary form. The size of the instance is $n + |u| + |v| + |w| + \| A \|_0 + \| R \|_0$.

Question. Is $(u^{-1}vu)w \stackrel{k}{\longleftrightarrow} w(u^{-1}vu)$ in G for $k \leq n$? ($u^{-1}vu$ is called a conjugation of u on v, denoted by $u \diamond v$.)

Distribution. Randomly and independently select positive integers l, m, n, and binary strings u, v, and w. Then randomly and independently choose binary strings $a_1, ..., a_l$ and $r_1, ..., r_m$. The random choices are made with respect to the default uniform probability distributions on positive integers and binary strings. Hence, the probability distribution is proportional to

$$\frac{2^{-(\|A \cup R\|_0 + |u| + |v| + |w|)}}{(lmn|u||v||w| \ \|A \cup R\|_1)^2}.$$

The distributional word problem for finitely presented groups was shown to be average-case NP-complete by Wang [Wan95a].

3.6 Distributional Halting Problem (Version 2)

This version of halting problem is with a flat distribution.

Distributional Halting Problem (Version 2)

Instance. Binary strings i, x, and t, where i is a positive integer. The size of the instance is $|i| + |x| + |t|$.

Question. Does M_i accept x within $|t|$ steps?

Distribution. Randomly select binary strings i, x, and t with respect to the default uniform distribution. Hence, the probability distribution is proportional to $2^{-(l+m+n)}l^{-2}m^{-2}n^{-2}$, where $l = |i|$, $m = |x|$, and $n = |t|$.

This version of the distributional halting problem was first shown to be average-case NP-complete by Gurevich [Gur91]. Randomized reductions are used to handle the "flatness." The version presented is from [Wan96].

3.7 Distributional Graph Edge Coloring Problem

We consider directed graphs in which nodes are labeled and may have self-loops. Let G be such a directed graph. An edge of G may be colored or left blank (*i.e.*, uncolored). A *spot* in a colored digraph is a 3-node subgraph with induced colored edges (including self-loops if there are any) and the nodes unlabeled. We only use a constant number of colors.[3] So it is easy to see that there are only constant number of different spots. The *coloration* $C(G)$ of G consists of the set of all spots induced from the colored graph G, and the number of blank edges. We will also specify a set R of constraints that specify what edges may be colored blank. For instance, one may specify that "only edges on a

[3]What would be the minimum number of colors that can make the distributional graph edge coloring problem to be complete for DistNP is an interesting issue. It was claimed that 20 colors [VL88], or even much less colors [Ven91], were sufficient. Here we would allow more colors to simplify technical requirements in the completeness proof, which will be presented in Section 6.3. Our main purpose here is to demonstrate that there is a graph problem with a flat distribution that is complete for DistNP.

Hamiltonian path may be colored blank."

DISTRIBUTIONAL GRAPH EDGE COLORING PROBLEM

Instance. A directed graph G of n nodes, a set C of spots, a positive integer k specifying the number of blank edges, and a set R of constraints specifying what edges may be colored blank. The size of the instance is k plus the size of G plus the size of C plus the size of R, where R is encoded as a binary string.

Question. Can G be colored, under the constraints in R, such that $C(G) = (C', k)$ and $C' \subseteq C$? (If so, we then say that G is colorable.)

Distribution. First choose a positive integer n with respect to the default uniform distribution. Then randomly and independently choose a directed graph of n nodes with uniform probability $4^{-\binom{n}{2}}2^{-n} = 2^{-n^2}$, a set of spots with probability $\Theta(1)$, the number of blank edges k with probability $\Theta(n^{-2})$, and the constraints R with probability $\Theta(|R|^{-2}2^{-|R|})$. Hence, the probability distribution is proportional to $n^{-4}2^{-n^2}|R|^{-2}2^{-|R|}$.

The distributional graph edge coloring problem was shown to be average-case NP-complete by Venkatesan and Levin [VL88] (see also [Ven91]) using a randomized reduction.[4]

3.8 Distributional Matrix Correspondence Problem

The distributional matrix correspondence problem concerns with 2×2 *unimodular* matrices, where a square matrix X is unimodular if all entries in X are integers and its determinant $\det(X) = 1$. For convenience, when we mention unimodular matrix in this paper we mean that its dimension is two-by-two. Let $\mathrm{SL}_2(\mathbb{Z})$ denote the set of unimodular matrices.

Define the *size* of a unimodular matrix X, denoted by $|X|$, to be the length of the binary representation of the maximal absolute value of its entries. It can be shown that the uniform probability of choosing X of size k among all positive unimodular matrices is $\Theta(k^{-2}2^{-2k})$.

DISTRIBUTIONAL MATRIX CORRESPONDENCE PROBLEM

Instance. A unimodular matrix A, a list $L(m) = \{(X_1, Y_1), \ldots, (X_m, Y_m)\}$ of unimodular matrices, and a positive integer n. The size of the instance is $n + |A| + \sum_{i=1}^{m}(|X_i| + |Y_i|)$.

Question. Are there unimodular matrices $X = X_{i_1}X_{i_2}\cdots X_{i_k}$ (by matrix multiplication) and $Y = Y_{i_1}Y_{i_2}\cdots Y_{i_k}$, where $k \leq n$ and $(X_{i_j}, Y_{i_j}) \in L(m)$, such that $AX = Y$? (If such X and Y exist, then we say that the instance has a solution of length at most n.)

Distribution. Randomly and independently choose positive integers n and m with respect to the default uniform distribution, then randomly and independently choose unimodular matrices $X_1, Y_1, \ldots, X_m, Y_m$. Hence, the probability

[4] Actually, the distributional graph edge coloring problem stated here is somewhat different from that in [VL88, Ven91]. But the proof technique is essentially the same.

distribution is proportional to

$$\left(\prod_{i=1}^{m} \frac{2^{-2(|X_i|+|Y_i|)}}{|X_i|^2|Y_i|^2}\right)\left(\frac{2^{-2|A|}}{|A|^2}\right)\left(\frac{1}{m^2}\right)\left(\frac{1}{n^2}\right).$$

In the definition of the distributional matrix correspondence problem, if we only allow positive unimodular matrices, where by "positive" we mean that every entry is a positive integer, then we obtain the DISTRIBUTIONAL POSITIVE MATRIX CORRESPONDENCE PROBLEM.

The distributional matrix correspondence problem was shown to be average-case NP-complete by Gurevich [Gur90] under a randomized reduction (see also [BG95]).

3.9 Distributional Matrix Transformation Problem

The distributional matrix transformation problem concerns with linear transformations on unimodular matrices. A *linear transformation* of $\mathrm{SL}_2(\mathbb{Z})$ is a function $T : \mathrm{SL}_2(\mathbb{Z}) \to \mathrm{SL}_2(\mathbb{Z})$ such that $T(\sum X_i) = \sum T(X_i)$ whenever all the X_i and $\sum X_i$ are unimodular matrices. (Note that $\mathrm{SL}_2(\mathbb{Z})$ is not closed under addition in general.) A linear transformation of $\mathrm{SL}_2(\mathbb{Z})$ can be represented by a 4×4 integer matrix, and it is decidable in polynomial time whether a given 4×4 integer matrix represents a linear transformation of $\mathrm{SL}_2(\mathbb{Z})$ [BG95].

Let T be a linear transformation and let $M(T)$ be its four-by-four integer matrix representation. Define the *size* of T to be the length of the largest absolute value (in the binary notation) of entries in $M(T)$. It can be shown that the uniform distribution of T among all linear transformations of size l is $\Theta(l^{-1}2^{-2l})$.

DISTRIBUTIONAL MATRIX TRANSFORMATION PROBLEM

Instance. A unimodular matrix X, a finite set S of linear transformation of unimodular matrices, and a natural number n. The size of the instance is n plus the size of X plus the sum of the sizes of all members in S.

Question. Does a linear transformation T exist, where $T = T_1 \circ T_2 \circ \cdots \circ T_k$, $k \leq n$, $T_i \in S$, such that $T(X)$ is the identity matrix?

Distribution. The three components are chosen randomly and independently. The integer component n is chosen with respect to the default uniform distribution $1/n^2$. The unimodular component X is chosen with probability $|X|^{-2}2^{-2|X|}$. Linear transformations are chosen with respect to the uniform distribution on transformations of the same size. Finally, the probability of S is proportional to the product of the probabilities of the members in S.

The distributional matrix transformation problem was shown to be average-case NP-complete by Gurevich [Gur90] and a full proof was published by Blass and Gurevich [BG95] using a randomized reduction.

3.10 Distributional Matrix Representability Problem

We consider square matrices with integer entries. The matrix representability problem is to decide, for a given matrix A and a set of matrices $\mathsf{M} = \{A_1, \ldots A_k\}$, whether A can be expressed as a product of matrices in M. This problem was shown to be undecidable for 6×6 matrices by Markov [Mar54] and was later improved to 4×4 matrices [Mar58]. Venkatesan and Rajagopalan [VR92] considered bounded version of this problem for 20×20 matrices as follows. The size of a matrix $A = (a_{ij})_{n \times n}$, denoted by $|A|$, is equal to $\sum_{i=1}^{n} \sum_{j=1}^{n} |a_{ij}|$, where a_{ij} is in binary form. Denote by M^m the set of all products of k matrices from M for $k \leq m$.

Distributional Matrix Representability Problem

Instance. Matrix A, a set of matrices $\mathsf{M} = \{A_1, \ldots A_k\}$, and a positive integer n. All matrices are 20×20 matrices with integer entries. The size of the instance is $n + |A| + \sum_{i=1}^{k} |A_i|$.

Question. Is $A \in \mathsf{M}^n$?

Distribution. Randomly and independently choose integer n, k, and all the entries in the matrices with respect to the default uniform distributions on integers and binary strings.

The distributional matrix representability problem was shown to be average-case NP-complete by Venkatesan and Rajagopalan [VR92].

It is easy to see that the above distributional problems are all in DistNP by noting that the size of a positive integer n is n itself (we may use a unary notation such as 1^n to denote n for this purpose).

4 Completeness Proofs (Part I)

We present in this section the completeness proofs for the distributional halting problem (version 1) and the distributional tiling problem, which are the simplest completeness proofs.

4.1 Distributional Halting (Version 1)

The completeness proof for the distributional halting problem (version 1), was first given by Gurevich [Gur91]. An easier and more comprehensible proof was given in [WB95] (see also [Wan96]), which will be presented here. Denote by K_1 the set of all positive instances of the halting problem and μ_{K_1} the probability distribution on (positive and negative) instances (i, x, n). Recall that the size of this instance is $|i| + |x| + n$.

Theorem 4.1 *The distributional halting problem (version 1) is complete for* DistNP.

Proof. Let (D, μ) be an arbitrary problem in DistNP. Then there is a nondeterministic Turing machine M which accepts D in polynomial time. By the

Distribution Controlling Lemma, there is a total, one-one, p-time computable and p-time invertible function α such that $\mu(x) \preceq 2^{-|\alpha(x)|}$. Define a Turing machine M' as follows. On input y, if $\alpha^{-1}(y)$ is defined, then M' simulates M on $\alpha^{-1}(y)$ and rejects otherwise. Clearly for all x, M accepts x iff M' accepts $\alpha(x)$. It is easy to see that M' on input $\alpha(x)$ is bounded in time $p(|x|)$ for some polynomial p. Let i be an index such that $M' = M_i$. Let $f(x) = (i, \alpha(x), p(|x|))$. Then f is one-one and p-time computable. By construction, $x \in D$ iff $f(x) \in K_1$. Moreover, $\mu_{K_1}(f(x)) = 2^{-|\alpha(x)|}/q(|x|)$ for some polynomial q, which dominates $\mu(x)$. It follows from Lemma 2.1 that $\mu \preceq_f \mu_{K_1}$. ∎

4.2 Distributional Tiling

The distributional tiling problem was the first problem known to be average-case NP-complete, which was shown by Levin [Lev86]. Gurevich provided a more accessible proof with all the details [Gur91]. The proof presented here is from [BW93].

Denote by BT the set of all positive instances (T, n, S) of the tiling problem and μ_{BT} the probability distribution on instance (T, n, S) (positive or negative). The idea of the proof is as follow. Let $(D, \mu) \in$ DistNP and let α be as in the proof of Theorem 4.1. Let $x' = \alpha(x)$. One might be tempted to reduce (D, μ) to $(\mathrm{BT}, \mu_{\mathrm{BT}})$ by reducing an instance x of D to an instance (T, n, S), where S represents the initial instantaneous description of an appropriate Turing machine on x' followed by some "blank" tiles. The problem with this approach is that $\mathbf{Pr}[S]$, which is at most $3^{-|x'|}$, is too small to dominate $\mu(x)$. If S does not end with "blank" tiles, then $\mathbf{Pr}[S]$ could be dominated by $2^{-|x'|}$. But something needs to be done to make sure that reductions will not reduce a negative instance of D to a positive instance of BT. This is done by a careful coding method. We first consider tiles with marked edges.

Theorem 4.2 *The distributional tiling problem is complete for* DistNP.

Proof. Let $(D, \mu) \in$ DistNP. It follows that there is a (non-deterministic) Turing machine M that accepts D in polynomial time. Without loss of generality, we assume that all the computation paths of M are bounded by a polynomial of input length. By the Distribution Controlling Lemma, there is a total, one-one, p-time computable and p-time invertible function α such that for all x, $\mu(x) \preceq 2^{-|\alpha(x)|}$.

We would like a Turing machine to be able to find the string $\alpha(x)$ inside a longer string. This can be done in the following way: for any string w, let σ be $|w|$ written in binary. Set $w_L = 0^{|\sigma|}1\sigma w$. Then given a string which has w_L as a prefix, M' can count the number of 0's before the initial 1, use this number to find the number σ, and then use σ to find w. Notice that $|w_L| = |w| + O(\log |w|)$. Let M' be a Turing machine which, given an input z beginning with w_L, determines w as above. (We can assume that if z does not begin with a 0 or that if a w cannot be determined because z is too short, then M' rejects z.) M' then determines whether $w = \alpha(x)$ for some x. This can be done in time polynomial in $|x|$. If so, M' then simulates M on x,

otherwise M' rejects. For convenience, let $x' = \alpha(x)$. Notice that for any given x, M' will either accept all or reject all inputs beginning with x'_L (depending on whether or not M accepts x), and all the computation paths of M' will be less than $p(|x|)$ for some polynomial p. For simplicity, we say that M' accepts or rejects x'_L. The fact that M' ignores any input after x'_L is important in the construction of our reduction. Clearly, M accepts x iff M' accepts x'_L.

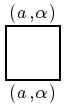
Let Q be the set of states of M', with q_0 the initial state of M', a the accepting state, and r the rejecting state. Let δ be the transition function of M', and B be the blank symbol. Let $T(M')$ consist of the following legal tiles:

for $\alpha \in \{0, 1, B\}$:

for $\alpha, \beta, \gamma \in \{0, 1, B\}, q \in Q - \{a, r\}, p \in Q$, $\delta(q, \alpha) = (p, \beta, R)$:

for $\alpha, \beta, \gamma \in \{0, 1, B\}, q \in Q - \{a, r\}, p \in Q$, $\delta(q, \alpha) = (p, \beta, L)$:

for $\alpha \in \{0, 1, B\}$:



and for $\alpha \in \{0, 1\}$:

The desired reduction is defined by $f(x) = (T(M'), p(|x|), S)$, where if x'_L is written $x'_{L,1}x'_{L,2}\ldots x'_{L,k}$ for $x'_{L,i} \in \{0, 1\}$, S consists of the tiles:

Note that the probability of the ith tile of S for $i = 2, \ldots, k = |x'_L|$ is $1/2$.

Clearly, f is one-one and p-time computable.

If S extends to a set of legal tiles which tile a $p(|x|) \times p(|x|)$ square, then S will have to occupy the bottom left hand side of the square. In this case, the symbols at the top of the bottom row will represent the initial ID of M' on some input beginning with x'_L, and the symbols at the top of the ith row from the bottom will represent the ID of a position reachable by M' within i steps. Since M' will reach the accepting or rejecting state in less than $p(|x|)$ steps,

and the tiling can only duplicate the accepting state, S can extend to a tiling only if M' accepts x'_L, which happens exactly when M accepts x. Similarly, if M' accepts x'_L, then S can be extended to a tiling of a $p(|x|) \times p(|x|)$ square.

We now verify that the reduction satisfies the domination requirement. To see this, we note that $\mu_{\mathrm{BT}}(f(x)) = \mu_{\mathrm{BT}}(T(M'), p(|x|), S)$, which is proportional to

$$\frac{1}{(p|x|)^3} \cdot \frac{1}{2^{|x'_L|}} = \frac{1}{(p(|x|)^3} \cdot \frac{1}{2^{|x'|+O(\log|x'|)}} = \frac{1}{q(|x|)} \cdot \frac{1}{2^{|x'|}}$$

for some polynomial q. Hence, $\mu \preceq \mu_{\mathrm{BT}} \circ f$. It follows by Lemma 2.1 that $\mu \preceq_f \mu_{\mathrm{BT}}$. ∎

It is easy to modify the above completeness proof to show that the distributional tiling problem with marked corners is also complete for DistNP. We do so by constructing tiles such that a tile letter represents a cell in an ID of M' that either encodes an input symbol and the direction to the head, or encodes a state of the head and the direction of the cell that the head looks at. Hence, there are only two different tiles that can be chosen for the initial sequence S of tiles after the first tile was selected, and so the domination property is satisfied.

5 Completeness Proofs (Part II)

A difficulty among completeness proofs for the average case complexity is that it is hard to maintain the domination property for probability distributions. In general, if the size of the output of a reduction is a linear growth of the size of its input, then it may damage the domination property. For example, if we were to reduce the halting problems to a problem (D, μ) in an effort to show that (D, μ) is complete, we would try to find a reduction f such that $\mu_K \preceq_f \mu$. If the reduction f transforms (i, x, n) to z such that z has a parameter y with $|y| \geq c|x|$ for some $c > 1$ and $2^{-|y|}$ being an irreducible factor of μ, then μ cannot dominate μ_K with respect to f. The only possibility left in this connection is for y to satisfy $|y| = |x| + O(\log|x|)$. A dynamic binary coding scheme was introduced by Gurevich [Gur91] to meet this requirement, which will be used throughout this section.

5.1 Dynamic Coding Scheme

Let A be a finite alphabet with $|A| > 2$. Given a binary string x (we also assume that x starts with 1), Gurevich [Gur91] designed a dynamic binary coding scheme to code A such that the following statements hold.

1. The length of each coded symbol is $O(\log|x|)$.
2. String x (not coded) is distinguishable from each coded symbol. In other words, no coded symbol can be a substring of x.
3. If a nonempty suffix z of a coded symbol u is a prefix of a coded symbol v, then $z = u = v$.

4. x can be written as a unique concatenation of the following fixed binary strings 1, 10, 000, 100, which are not prefixes of any coded symbol.

The four strings 1, 10, 000, 100 are called *base strings* [Wan95a]. Gurevich [Gur91] originally used strings 1, 10, 00, and 000 as base strings, which, in general, does not form a unique concatenation for strings beginning with 1.

The construction of such a coding system can be done as follows. Let R be the regular set $0100(00+11)^*11$. Let l be the least even integer such that $2^{(l-6)/2} \geq |x| + |A|$. Hence, $l = O(\log |x|)$. The string x has at most $|x|$ substrings of length l. So we can select a set S of R-strings of length l such that $|S| = |A|$, no string in S is a substring of x, and every string in S starts with 01. Moreover, from R it is straightforward to show that the third condition also holds [Gur91]. To see that the fourth condition is satisfied, let y be a string which does not start with 01 and is different from 0, then it is straightforward to show by induction that y forms a unique concatenation from the base strings. Note that the last two conditions enable coding x without recoding the 0's and 1's in other already coded symbols.

So we can use S to code A by assigning one element in S to represent one symbol in A, and this dynamic coding scheme satisfies all of the above four conditions.

5.2 Distributional Post Correspondence

The distributional Post correspondence problem was shown to be complete for DistNP by Gurevich [Gur91]. A simplified proof was given in [WB95] and will be presented here. For convenience, for a list $L = \langle (u_1, v_1), ..., (u_m, v_m) \rangle$, we call $\{u_1, ..., u_m\}$ the A-list of L, and $\{v_1, ..., v_m\}$ the B-list of L. Let C denote the set of all positive instances of the problem. Recall that μ_C denotes the uniform probability distribution on instances (L, n) of the bounded Post correspondence problem.

A negative instance $(L(m), n)$ does not prevent $L(m)$ from having a solution of length greater than n. We say that an instance $(L(m), n)$ is *robust* if either $L(m)$ has a solution of length at most n or $L(m)$ does not have solutions at all.

Theorem 5.1 *The distributional Post correspondence problem is complete for* DistNP.

Proof. For simplicity, we use the standard nondeterministic one-tape single-headed Turing machines as our computation model. The tape is bounded on the left and unbounded to the right. At the initial state, the input is left justified on the tape and the head is positioned at the leftmost symbol of the input.

Let (D, μ) be a distributional problem in DistNP. Then there is a (nondeterministic) Turing machine M which accepts D in polynomial time. Without loss of generality, we assume that all the computation paths of M are bounded by a polynomial of the length of inputs. By the Distribution Controlling Lemma , there is a total, one-one, p-time computable and p-time invertible function α such that $\mu(x) \preceq 2^{-|\alpha(x)|}$ for all x. Let $x' = \alpha(x)$.

We now construct a Turing machine M' with only one halting state. M' takes input $1w$ and determines whether $w = \alpha(x)$ for some x. If $\alpha^{-1}(w)$ is not defined, then M' goes into an infinite loop state, otherwise, M' simulates M on x. M' halts when M on x reaches an accepting state. If M on x reaches a rejecting state, then M' simply goes into an infinite loop state.

It is easy to see that for all x, the length of each ID of M' on input $1x'$ is bounded by a fixed polynomial of $|x|$, and M' on input $1x'$ halts (has a halting computation) if and only M accepts x. Moreover, M' on input $1x'$ halts if and only if it halts within $p(|x|)$ steps for some fixed polynomial p.

Let h be the halting state of M', q_0 the initial state of M', B the blank symbol, Q the set of states of M', and δ the transition function of M'. Let $z = 1x'$. Let $\Sigma' = Q \cup \Sigma \cup \{B, \$, \&, !\}$. Fix a dynamic binary coding system for z and the finite set Σ'. We know that the length l of a coded symbol is $O(\log |x'|) = O(\log |x|)$. For any word w on Σ', we use $\underline{w}$ to denote its coded word of w. So $\underline{z}$ is the string obtained from z by replacing 0 with $\underline{0}$ and 1 with $\underline{1}$.

Let $L = L(z)$ be an instance of the distributional Post correspondence problem comprising the following pairs of binary strings.

1. (Group I) $(\underline{\$}, \underline{\$z\&!q_0})$.
2. (Group II) $(u, \underline{u'})$ for $u \in \{1, 10, 000, 100\}$, and $\underline{u'}$ is obtained from u by replacing 0 with $\underline{!0}$ and 1 with $\underline{!1}$.
3. (Group III) $(\underline{X!},\ \underline{!X})$ for each $X \in \Sigma \cup \{B, \&\}$.
4. (Group IV) For each $q \in Q - \{h\}$ and $X, Y, Z \in \Sigma \cup \{B\}$,

$$\begin{array}{ll}
(\underline{q!X!},\ \underline{!Y!p}) & \text{if } \delta(q, X) = (p, Y, R), \\
(\underline{Z!q!X!},\ \underline{!p!Z!Y}) & \text{if } \delta(q, X) = (p, Y, L), \\
(\underline{q!\&!},\ \underline{!Y!p!\&}) & \text{if } \delta(q, B) = (p, Y, R), \\
(\underline{Z!q!\&!},\ \underline{!p!Z!Y!\&}) & \text{if } \delta(q, B) = (p, Y, L).
\end{array}$$

5. (Group V) $(\underline{X!h!},\ \underline{!h})$, $(\underline{X!h!Y!},\ \underline{!h})$, $(\underline{h!Y!},\ \underline{!h})$ for $X, Y \in \Sigma \cup \{B\}$.
6. (Group VI) $(\underline{h!\&!\&},\ \underline{!\&})$.

A pair (u, v) is a *partial solution* to the distributional Post correspondence problem with list L if u is a prefix of v, and u and v are the concatenation of corresponding strings of A-list and B-list respectively. If $uw = v$, then call w the remainder of (u, v).

Let $\underline{y}$ be a string of coded symbols in $\Sigma \cup \{B\}$. Denote by $\underline{\tilde{y}}$ the string obtained from $\underline{y}$ by replacing each coded symbol $\underline{X}$ with $\underline{X!}$ and with the last $\underline{!}$ omitted. Suppose from initial ID $q_0 z$ there is a valid sequence of k more ID's. Let $\alpha_i q_i \beta_i$ denote an ID in a usual way. We claim that there is a partial solution

$$\begin{aligned}(u, v) = (&\underline{\$z\&!q_0!\tilde{z}!\&!\tilde{\alpha}_1!q_1!\tilde{\beta}_1!\&!\cdots\&!\tilde{\alpha}_{k-1}!q_{k-1}!\tilde{\beta}_{k-1}!\&!},\\ &\underline{\$z\&!q_0!\tilde{z}!\&!\tilde{\alpha}_1!q_1!\tilde{\beta}_1!\&\cdots!\&!\tilde{\alpha}_k!q_k!\tilde{\beta}_k!\&}).\end{aligned}$$

Moreover, this is the only type of partial solution whose larger string is as long as $|v|$. Since the length of each ID is bounded by a fixed polynomial of $|x|$, it is a partial solution of length bounded by k times a polynomial of $|x|$.

It is obvious that any solution must start with $(\underline{\$}, \underline{\$z\&!q_0})$. To continue, the only choice now is from pairs in Group II because of the coding system we used. This gives the correspondence of z to $\underline{!\tilde{z}}$. Then the only pair that can be applied is $(\underline{\&!}, \underline{!\&})$. This gives a partial solution $(\underline{\$z\&!}, \underline{\$z\&!q_0!\tilde{z}!\&})$. So this verifies the case when $k = 0$. By a simple induction and the coding scheme we used, one can easily prove the statement. For completeness, we present its proof below.

Suppose that the statement is true for some k and that $q_k \neq h$. We can easily show that it is true for $k+1$. The remainder of the pair (u, v) is $w = \underline{\tilde{\alpha}_k!q_k!\tilde{\beta}_k!\&}$. The next pairs must be chosen so that their strings from the A-list form w. No matter what symbols appear to the right and left of q_k, it will take a pair from Group IV to enable the partial solution to be continued past q_k. This pair represents, in a natural way, a move of M' from ID $\alpha_k q_k \beta_k$. Use choices from Group III for other symbols of w. No other choices will enable w to be composed of elements in the A-list. We can thus obtain a new partial solution $(\underline{v!}, \underline{v!\tilde{\alpha}_{k+1}!q_{k+1}!\tilde{\beta}_{k+1}!\&})$. It is straightforward to see that $\alpha_{k+1}q_{k+1}\beta_{k+1}$ is an ID that M' can reach on one move from $\alpha_k q_k \beta_k$. Also, there are no other partial solutions whose length of the second string equals $|\underline{v!\tilde{\alpha}_{k+1}!q_{k+1}!\tilde{\beta}_{k+1}!\&}|$. Since each ID is bounded by a polynomial of $|x|$, the length of this partial solution is bounded by k times a polynomial of $|x|$. In addition, if $q_k = h$, then k is bounded by a polynomial of $|x|$. It is easy to find pairs from Groups III and V which, when preceded by the partial solution (u, v) and followed by the pair in Group VI, provide a solution of length bounded by a polynomial of $|x|$ to $L(z)$. If M' does not reach the halting state, no pairs from Groups V and VI may be used. Therefore, there may be partial solutions, but the second string must exceed the first string, so no solution is possible.

We conclude that $L(z)$ has solution if and only if M' on input z halts. We know that M' on input z halts if and only if M' on input z halts within $p(|x|)$ steps. So if $L(z)$ has a solution, then it has a solution bounded by a polynomial of $|x|$. So there is a polynomial q such that M' on input z halts if and only $L(z)$ has a solution of length at most $q(|x|)$.

Let $f(x) = (L(z), q(|x|))$. It is easy to see that $f(x)$ is a robust instance. Notice that $z = 1x'$. Clearly, f is one-one and p-time computable. Moreover, D is reducible to C via f. Now we verify that $\mu \preceq \mu_C \circ f$. Since $|\underline{\$z\&!q_0}| = |x'| + O(l)$ and any other string in $L(1x')$ has length $O(l)$, where $l = O(\overline{\log |x|})$, it is easy to see that $\mu_C(f(x)) = 2^{-|x'|}/g(|x|)$ for some polynomial g. Hence, $\mu(x) \preceq \mu_C(f(x))$. ∎

In the proof of Theorem 5.1, if we let $L'(z)$ be a new list by omitting $(\underline{\$}, \underline{\$z\&!q_0})$ (Group I) from the list $L(z)$ and reversing the order of every remaining pair, then $x \in D$ iff $(\underline{z\&!q_0}, L'(z), q(|x|))$ is a positive instance of the distributional string correspondence problem. Moreover, it is straightforward to verify that the reduction defined by $f(x) = ((\underline{z\&!q_0}, L'(z), q(|x|))$ satisfies the domination requirement. This provides a proof to the following theorem.

Theorem 5.2 *The distributional string correspondence problem is complete for* DistNP.

Gurevich [Gur91] showed that the distributional palindrome problem is complete for DistNP by reducing a restriction of the distributional Post correspondence problem to the problem. An instance L of the Post correspondence problem is called *palindrome sensitive* if there is no palindrome $u_{i_1} \cdots u_{i_k} v_{i_k}^{-1} \cdots v_{i_1}^{-1}$ such that $|u_{i_1} \cdots u_{i_k}| \neq |v_{i_k}^{-1} \cdots v_{i_1}^{-1}|$, where (u_{i_j}, v_{i_j}) belongs to L. As a corollary of the proof of Theorem 5.1, it is straightforward to show the following lemma.

Lemma 5.3 *The restriction of the distributional Post correspondence problem to instances (L, n) such that L is palindrome sensitive is complete for* DistNP.

Theorem 5.4 *The distributional palindrome problem is complete for* DistNP.

Proof. We will reduce the palindrome sensitive version of the distributional Post correspondence problem to the distributional palindrome problem. Given a palindrome sensitive instance $L = \langle (u_1, v_1), \ldots, (u_m, v_m) \rangle$ and a positive integer n, the desired reduction produces a grammar G with productions $T \to u_1 T v_1 \mid \cdots \mid u_m T v_m \mid e$, and number $n+1$. The domination requirement is obvious. We need to check that L has a solution of length $\leq n$ iff G produces a nonempty palindrome in at most $n+1$ steps.

It is clear that every solution $u_{F(1)} \cdots u_{F(k)} = v_{F(1)} \cdots v_{F(k)}$ for L gives rise to a $(k+1)$-step derivation of the palindrome $u_{F(1)} \cdots u_{F(k)} v_{F(k)}^{-1} \cdots v_{F(1)}^{-1}$. Suppose that G produces a nonempty palindrome $u_{F(1)} \cdots u_{F(k)} v_{F(k)}^{-1} \cdots v_{F(1)}^{-1}$ in $n+1$ steps. Since L is palindrome sensitive, $u_{F(1)} \cdots u_{F(k)} = v_{F(1)} \cdots v_{F(k)}$. This completes the proof. ∎

5.3 Distributional String-Rewriting

The distributional word problem for Thue systems was shown to be complete for DistNP by Wang and Belanger [WB95]. Recall that μ_W denotes the probability distribution on instances of the distributional word problem for Thue systems.

Theorem 5.5 *The distributional word problem for Thue systems is complete for* DistNP.

Proof. We use the standard nondeterministic one-tape single-headed Turing machines as in the proof of Theorem 5.1. Let (D, μ) be a distributional problem in DistNP. Then there is a (nondeterministic) Turing machine M that accepts D in polynomial time. Without loss of generality, we assume that all the computation paths of M are bounded by a polynomial of the length of inputs. By the Distributional Controlling Lemma, there is a total, one-one, p-time computable and p-time invertible function α such that $\mu_(x) \preceq 2^{-|\alpha(x)|}$ for all x. Let $x' = \alpha(x)$.

We now construct a Turing machine M' with only one halting state. M' takes input $1w$ and determines whether $w = \alpha(x)$ for some x. This can be

done in time polynomial in $|x|$. If $\alpha^{-1}(w)$ is not defined, then M' goes into an infinite loop state, otherwise, M' simulates M on x. If M on x reaches an accepting state, then M' erases all the tape symbols, moves the head to the left, and halts. If M on x reaches a rejecting state, then M' simply goes into an infinite loop.

Let $x' = \alpha(x)$. Clearly for all x, M' on input $1\alpha(x)$ has a halting computation if and only if M accepts x. Because of the time bound on M, it is easy to see that if M' on input $1x'$ halts, then the number of steps $g(1x')$ it takes is bounded by a polynomial of $|x|$.

Now we construct a set $\Pi = \Pi(M')$ of productions to describe instantaneous descriptions (ID's) M'. Let h be the halting state of M', s the initial state of M', B the blank symbol, Q the set of states of M', and δ the transition function of M'. We use a symbol \$ as an end marker to handle the blank tape detail. Π consists of the following ordered pairs (productions):

1. For any $q \in Q - \{h\}$, $p \in Q$, $a, b, c \in \Sigma \cup \{B\}$, if $\delta(q, a) = (p, b, R)$, then Π contains $\langle qac, bpc\rangle$, and $\langle qa\$, bpB\$\rangle$.
2. For any $q \in Q - \{h\}$, $p \in Q$, $a, b, d, \in \Sigma \cup \{B\}$, and $c \in \Sigma \cup \{\$\}$, if $\delta(q, a) = (p, b, L)$, then Π contains $\langle dqB\$, pd\$\rangle$, and $\langle dqac, pdbc\rangle$ for $a \neq B$, $c \neq \$$ or $b \neq B$.

We can now construct a Thue system $\Gamma = \Gamma(M')$. Let $\Sigma_\Gamma = Q \cup \Sigma \cup \{B, \$\} \cup \{r_i : i \in \Pi\}$, where the r_i are new symbols. We use these new symbols to handle nondeterminism in case the inverse of a production is applied. Let $z = 1x'$, and fix a dynamic binary coding scheme for z and the finite set Σ_Γ. We know that the length l of a coded symbol is $O(\log |x'|) = O(\log |x|)$. As before, for any word w on Σ_Γ, we use $\underline{w}$ to denote the coded form of w.

Γ will consist of the following (unordered) pairs:

1. (Group I) For each $a \in \{s, 0, 1\}$ and each $u \in \{1, 00, 000, 100\}$, Γ includes $(\underline{a}\underline{u}, \underline{au})$.
2. (Group II) For each $i = \langle \alpha, \beta\rangle \in \Pi$, Γ includes $(\underline{\alpha}, \underline{r_i\beta})$.
3. (Group III) For each $a \in \Sigma$, $i \in \Pi$, Γ includes $(\underline{ar_i}, \underline{r_i a})$.
4. (Group IV) For each $i = \langle \alpha, \beta\rangle \in \Pi$, Γ includes $(\underline{r_i h}, \underline{h})$.

Now consider strings $\underline{\$sz\$}$ and $\underline{\$hB\$}$. From conditions 3 and 4 in the coding scheme, using the productions in Group I we can transform z into $\underline{z}$ without miscoding the already coded strings. Moreover, starting from string $\underline{\$sz\$}$, the only productions that can be applied are the ones from Group I because of condition 3 in the coding scheme, and the only way to do it (which will transforms z to $\underline{z}$ eventually) is to start from $\underline{s}$ and the leftmost symbol of z. Notice that when a production in Group I is applied, it must start from a coded symbol and the leftmost symbol of the rest of the non-coded string of z. Note that the productions in Group II may be applied before z has been completely transformed to $\underline{z}$. Let m be the number of times of applying productions in Group I to transform z to $\underline{z}$.

Assume that M halts on z. Using productions from Group I, $\underline{\$sz\$}$ can be transformed into $\underline{\$sz\$}$ in m steps. Then, using the productions from Group II

that duplicate the steps of M' that will halt on z, and using the productions from Group III as necessary, $\underline{\$sz\$}$ can be transformed into $\underline{\$r_{i_1}r_{i_2}\ldots r_{i_{g(z)}}hB\$}$. After each application of a Group II production, it will be necessary to apply at most $g(z)$ Group III productions to move the symbol $\underline{r_i}$ as far left as possible. So transforming $\underline{\$sz\$}$ into $\underline{\$r_{i_1}r_{i_2}\ldots r_{i_{g(z)}}hB\$}$ will take at most $g(z)+(g(z))^2$ steps. Finally, $\underline{\$r_{i_1}r_{i_2}\ldots r_{i_{g(z)}}hB\$}$ can be transformed into $\underline{\$hB\$}$ in $g(z)$ steps by Group IV productions.

Conversely, if $\underline{\$sz\$}$ can be transformed into $\underline{\$hB\$}$, then M' will halt on z. It is important to note that the productions from Group II may undo a simulated step of M', but since each simulated step is kept track of through the $\underline{r_i}$'s, it can only bring the string back to a previous simulated ID of M'.

So we have M' halts on input z if and only if $\underline{\$sz\$} =_\Gamma^{m+2g(z)+(g(z))^2} \underline{\$hB\$}$.

Let $f(x) = (\Gamma, \underline{\$sz\$}, \underline{\$hB\$}, m+2g(z)+(g(z))^2)$. Then clearly f is one-one, and p-time computable. We also know that M accepts x if and only if M' halts on z in time $g(z)$ if and only if $f(x) \in W$.

Now we verify that $\mu \preceq \mu_W \circ f$. Notice that the length of each production in Γ is bounded by $O(l)$, and the number of productions in Γ is independent of input x. Notice also that $|\underline{\$sz\$}| = |x'| + O(l)$, $|\underline{\$hB\$}| = O(l)$, and $m+2g(z)+(g(z))^2 \leq p(|x|)$ for some polynomial p. Since $l = O(\log|x|)$, $2^{-O(l)}$ is bounded by a polynomial of $|x|$. So there is a polynomial q such that $\mu_W(f(x)) = 2^{-|x'|}/q(|x|)$. Hence, $\mu(x) \preceq \mu_W(f(x))$. This completes the proof. ∎

To show that the distributional common ancestor and descendant problems are complete for DistNP, we first consider a restricted version of the word problem in which n is required to be an even integer as part of the instance. Clearly, the proof of Theorem 5.5 carries out easily to this restricted version as we can simply select the polynomial p to be even. Hence, the restricted distributional word problem for Thue systems is complete for DistNP. The desired reduction from the restricted distributional word problem for Thue systems to the distributional common ancestor problem is defined by $f(\Pi, u, v, n) = (\Pi, u, v, n/2)$. Since the rewriting rules in Π can go both ways, it is easy to see that (Π, u, v, n) is a positive instance of the distributional word problem iff $(\Pi, u, v, n/2)$ is a positive instance of the distributional common ancestor problem. The desired property of domination for distributions is obviously satisfied. The very same reduction applies to the distributional common descendant problem. This proves the following result [Wan95b].

Theorem 5.6 *The distributional common ancestor problem and the distributional common descendant problem are complete for* DistNP.

5.4 Distributional Word Problem for Groups

The distributional word problem for finitely presented groups was shown to be complete for DistNP by Wang [Wan95a]. The idea of proving this result is as follows. Given a randomized NP decision problem $(D, \mu) \in$ DistNP, we construct a reduction such that for any instance x of D, the reduction transforms x to a finitely presented group $G = [A; R]$, strings u, v, w, and

a unary notation for $p(|x|)$ with the following properties, where p is a fixed polynomial.

1. $(u^{-1}vu)w$ is equivalent to $w(u^{-1}vu)$ in G iff $(u^{-1}vu)w \stackrel{k}{\longleftrightarrow} w(u^{-1}vu)$ in G for $k \leq p(|x|)$ iff $x \in D$.
2. μ is dominated by the default uniform distribution of the word problem with respect to the reduction.

We will first construct an intermediate word problem for Thue systems. We will then construct the desired finitely presented groups using the HNN (Higman-Neumann-Neumann) extensions. A dynamic coding scheme is used to make sure that the domination property is satisfied.

Theorem 5.7 *The distributional word problem for finitely presented groups is complete for* DistNP.

Proof. For simplicity, we again use the standard nondeterministic one-tape single-headed Turing machines as our computation model as in the proof of Theorem 5.5. Let (D, μ) be an arbitrary distributional problem in DistNP. By the Distribution Controlling Lemma, there is a total, one-one, p-time computable and p-time invertible function σ such that $\mu(x) \preceq 2^{-|\sigma(x)|}$. Let M be a (nondeterministic) Turing machine that accepts D in polynomial time. Without loss of generality we assume that all the computation paths of M are bounded by a polynomial on the length of inputs.

We construct a Turing machine M' with only one halting state exactly the same as in the proof of Theorem 5.5. We assume that when M' enters an infinite loop, it does not change anything on the tape. Then for all x, M' on input $1\sigma(x)$ has a halting computation if and only if M accepts x. Because of the time bound on M, it is easy to see that if M' on input $1\sigma(x)$ halts, then the number of steps it takes is bounded by a polynomial on $|x|$. So we have

Lemma 5.8 *M accepts x in polynomial time iff M' halts on $1\sigma(x)$ in polynomial time in terms of $|x|$ iff M' halts on $1\sigma(x)$.*

We construct a set $\Pi = \Pi(M')$ of productions to describe instantaneous descriptions of M' exactly the same as in the proof of Theorem 5.5. We now construct a Thue system $\Gamma = \Gamma(M')$ based on Π, which is somewhat different from that in the proof of Theorem 5.5. We first construct a new set of symbols S_1 and a disjoint new set of states Q_1. We want Γ to contain productions only in the form of $EqF = HpK$, where p, q are states in Q_1, and E, F, H, K are strings on S_1. In so doing, our job will become relatively easier when constructing a finitely presented group that meets our needs.

Since Π is finite, we can fix an order $ord : \Pi \to \{1, 2, ..., |\Pi|\}$ for the ordered pairs in Π. We use new symbols d_i, $i = 1, ..., |\Pi|$ to handle nondeterminism in case the inverse of a production is applied. We use two extra states s_1 and s_2 to transform input $1\sigma(x)$ to its coded form before the actual simulation of M' on $1\sigma(x)$ begins. We use another two extra states s_3, s_4 and a set of new symbols $Q' = \{q' : q \in Q\}$ as markers to move d_i around when needed. Let

$$S_1 = \{0, 1, B, \$\} \cup \{d_i : i = 1, ..., |\Pi|\} \cup Q', \ Q_1 = Q \cup \{s_1, s_2, s_3, s_4\}.$$

Γ will contain $\gamma = 23 + |Q| + 7|\Pi| + 3|\Pi||Q| + |\Pi|^2$ productions, which will be described later.

We then construct a finitely presented group $G = G(M')$ based on Γ. Let S_2 be a new set defined below.

$$S_2 = S_1 \cup Q_1 \cup \{\chi, \kappa, \tau\} \cup \{r_i : i = 1, ..., |\Gamma|\}.$$

Some extra symbols are needed for writing quantified relations. Let S_3 be a finite alphabet distinct from S_2 which is sufficient to describe quantified statements. For instance, S_3 may contain quantifier symbols, relational operation symbols, and variable symbols, etc.

Let $z = 1\sigma(x)$. Fix a dynamic binary coding scheme for z and the finite set

$$S_4 = S_2 \cup S_3 \cup \{!, \uparrow, (,)\}.$$

The length of any coded symbol is $O(\log|\sigma(x)|) = O(\log|x|)$. For any positive word w on S_4, we use $\underline{w}$ to denote the coded word of w.

Let $\mathcal{B} = \{1, 10, 100, 000\}$. The string-rewriting system Γ consists of the following productions:

Γ1: $\forall u \in \mathcal{B}$:

$$\begin{aligned} \underline{s}u &= \underline{\$us_1} \\ \underline{s_1}u &= \underline{us_1} \\ \underline{us_1\$} &= \underline{s_2u\$} \\ \underline{us_2} &= \underline{s_2u} \\ \underline{\$s_2} &= \underline{\$s} \end{aligned}$$

Γ2: $\forall\langle\alpha, \beta\rangle \in \Pi$: $\underline{\alpha} = \underline{d_i\beta}$, where $i = ord(\langle\alpha, \beta\rangle)$

Γ3: $\forall a \in \{0, 1\}, \forall q \in Q, \forall d_i, d_j$:

$$\begin{aligned} \underline{d_iq} &= \underline{d_is_3q'} \\ \underline{d_iaq} &= \underline{d_ias_3q'} \\ \underline{d_ias_3} &= \underline{d_is_3a} \\ \underline{ad_is_3} &= \underline{d_is_3a} \\ \underline{d_jd_is_3} &= \underline{d_jd_is_4} \\ \underline{\$d_is_3} &= \underline{\$d_is_4} \\ \underline{s_4a} &= \underline{as_4} \\ \underline{s_4q'} &= \underline{q} \end{aligned}$$

Γ4:

$$\begin{aligned} \forall d_i : \underline{d_ih} &= \underline{h} \\ \underline{\$hB\$} &= \underline{h} \end{aligned}$$

Lemma 5.9 *There is a polynomial q such that M' halts on input z iff $\underline{\$z\$} \overset{*}{\leftrightarrow} \underline{h}$ in Γ iff $\underline{\$z\$} \overset{k}{\longleftrightarrow} \underline{h}$ in Γ for $k \leq q(|x|)$. Moreover, the length of each word ever obtained in the transformation process is bounded by $q(|x|)$.*

Proof. Notice that starting from string $\underline{\$z\$}$, the only productions that can be applied are the ones in $\Gamma 1$ starting with the first one until $\underline{\$z\$}$ is transformed to $\underline{\$\$z\$}$. Because of the coding scheme we use this transformation can be done without miscoding the already coded strings. Since z can be uniquely written as $u_1 \cdots u_m$, where $u_i \in \mathcal{B}$, this transformation can be carried out in $2m+3$ steps.

Assume that M' halts on z in polynomial time $\theta(|x|)$. Using the rules in $\Gamma 2$, which duplicate the steps of M' that halts on x, and using the rules in $\Gamma 3$ as necessary, $\underline{\$\$z\$}$ can be transformed into $\underline{\$d_{i_1}d_{i_2}\ldots d_{i_{g(z)}}hB\$}$. After each application of a $\Gamma 2$ rule, it will be necessary to apply at most $g(z) \leq 2\theta(|x|)+3$ $\Gamma 3$ rules to move the symbol $\underline{d_i}$ as far left as possible. So transforming $\underline{\$\$z\$}$ into $\$d_{i_1}d_{i_2}\ldots d_{i_{g(z)}}hB\$$ will take at most $2\theta(|x|)(\theta(|x|)+2)$ steps. Finally, $\underline{\$d_{i_1}d_{i_2}\ldots d_{i_{g(z)}}hB\$}$ can be transformed into $\underline{h}$ in $\theta(|x|)+1$ steps by $\Gamma 4$ rules.

Conversely, if $\underline{\$z\$}$ can be transformed into $\underline{h}$, then M' will halt on x. It is important to note that the inverse of a production in Π can undo a simulated step of M', but since each simulated step is kept track of through the d_i's using the $\Gamma 2$ rules, it can only bring the string back to a previous simulated ID of M'.

Letting $q(n) = 2m + \theta(n)(2\theta(n)+5)$ completes the proof. ∎

We now construct a finitely presented group G. The set of generators of G is

$$A = \mathcal{B} \cup \{\underline{a} : a \in S_2\}.$$

The symbols $\underline{!}$ and $\underline{\uparrow}$ are used to represent negative and power words. The symbols in S_3 are used to represent quantified relations. We assume that all integers are written in the original binary form (*i.e.*, not coded in the dynamic coding scheme). Recall that each coded symbol is a string of the form 0100(00+11)*11. We code a positive binary integer by replacing 1 with 10, 0 with 01. A binary number so coded is easily distinguished from any coded symbol in S_4. Let bin(k) denote such a coded binary number.

We will use words of the form $\underline{a}^{2^n-1}$ in constructing group relations. Although it is exponential to write such a word as a direct concatenation of symbol a, it can be represented in a much compressed form without affecting the actual group operations. We represent negative words and power words as follows.

- $\underline{a}^{2^n}$ is coded by $\underline{a\uparrow}\text{bin}(n)$, denoted as $\underline{a^{2^n}}$, where n is positive.
- $\underline{a}$ is also denoted as $\underline{a^1}$.
- $\underline{a}^n$ is coded by $\underline{a^{2^{n_1}}a^{2^{n_2}}\cdots a^{2^{n_l}}}$, denoted as $\underline{a^n}$, where $n = 2^{n_1} + 2^{n_2} + \cdots + 2^{n_l}$, and $n_1 > n_2 > \cdots > n_l \geq 0$.
- $\underline{a}^{-2^n}$ is coded by $\underline{a!\uparrow}\text{bin}(n)$, denoted as $\underline{a^{-2^n}}$, where n is positive.
- $\underline{a}^{-1}$ is coded by $\underline{a!}$, denoted as $\underline{a^{-1}}$.

- $\underline{a^{-n}}$ is coded by $\underline{a^{-2^{n_1}}a^{-2^{n_2}}\cdots a^{-2^{n_l}}}$, denoted as $\underline{a^{-n}}$, where $n = 2^{n_1} + 2^{n_2} + \cdots + 2^{n_l}$, and $n_1 > n_2 > \cdots > n_l \geq 0$.

This coding scheme provides a much shorter representation for power words. For instance, the length of $\underline{a^{2^n}}$ is $O(\log|x| + \log n)$. Notice that this is only a representation of power words and so it does not introduce new generating symbols or relations. For any word w on S_4, we use $\underline{w}$ to denote the coded word of w. One can easily define operations for the power words so coded following the standard rules such that, for example, $\underline{a^{\pm m}a^{\pm n}} = \underline{a^{\pm m \pm n}}$, $(\underline{a^{\pm m}})^{-1} = \underline{a^{\mp m}}$, $(\underline{a^{\pm m}b^{\pm n}})^{-1} = \underline{b^{\mp n}a^{\mp m}}$. These operations can be carried out in polynomial time in terms of $\log|x|$, the length of m, and the length of n.

Let X be a word. Then $\underline{X^{\pm n}}$ is coded as above by replacing $\underline{a}$ with $\underline{(X)}$, denoted as $\underline{X^{\pm n}}$, where $n > 0$. If Y is a coded word already, then $\underline{Y^{\pm n}}$ is coded as above by replacing $\underline{a}$ with $\underline{(Y)}$. Group operations can be applied on this representation in a natural way. For example, $(X^{-1})^{-1} = X$, $(X_1X_2\cdots X_l)^{-1} = X_l^{-1}\cdots X_2^{-1}X_1^{-1}$, where X and X_i are coded words.

Let x and y be variables, we write $\underline{x^y}$ to denote the concatenation of $\underline{x}$ for y times. Symbolically, $\underline{x^y}$ is written as $\underline{x \uparrow y}$ and $\underline{x^{-y}}$ is written as $\underline{x! \uparrow y}$. When y is substituted with an actual value, the rules above are followed.

We now define a finite set of relations $R = R(M')$ to construct a finite presentation of group $G = [A; R]$.

Fix an order for the rewriting rules in Γ. Suppose $X \equiv a_1^{\epsilon_1}\cdots a_m^{\epsilon_m}$ is a (not necessarily positive) word, then $\overline{X} \equiv a_1^{-\epsilon_1}\cdots a_m^{-\epsilon_m}$.

R consists of the following relations:

$R1$: $\forall a \in S_1$: $\underline{\chi a} = \underline{a\chi^2}$.

$R2$: $\forall a \in S_1$ and $\forall r_i$: $\underline{r_i a} = \underline{a\chi r_i \chi}$.

$R3$: $\underline{r_i^{-1}\overline{E}qFr_i} = \underline{\overline{H}pK}$, if $\underline{EqF} = \underline{HpK}$ is the i-th relation in Γ in the fixed order. Here p and q are states in Q_1.

$R4$:

$$\begin{aligned} \underline{\tau\chi} &= \underline{\chi\tau} \\ \forall r_i: \ \underline{\tau r_i} &= \underline{r_i\tau} \end{aligned}$$

$R5$:

$$\begin{aligned} \underline{\kappa\chi} &= \underline{\chi\kappa} \\ \forall r_i: \ \underline{\kappa r_i} &= \underline{r_i\kappa} \\ \underline{\kappa(h^{-1}\tau h)} &= \underline{(h^{-1}\tau h)\kappa} \end{aligned}$$

$R6$: Let W be a variable representing positive words on S_1, let t be a variable representing $2^{|W|}$. For all W with length $\leq q(|x|)$ and for all r_i:

1.

$$\underline{r_i W} = \underline{W\chi^t\chi^{-1}r_i\chi^t\chi^{-1}}$$

$$\begin{aligned}
\underline{\overline{W} r_i^{-1}} &= \underline{\chi^{-t}\chi r_i^{-1}\chi^{-t}\chi\overline{W}} \\
\underline{r_i^{-1}W} &= \underline{W\chi^{-t}\chi r_i^{-1}\chi^{-t}\chi} \\
\underline{\overline{W} r_i} &= \underline{\chi^{t}\chi^{-1} r_i\chi^{t}\chi^{-1}\overline{W}}
\end{aligned}$$

2.

$$\begin{aligned}
\underline{\tau\chi^t\chi^{-1}} &= \underline{\chi^t\chi^{-1}\tau} \\
\underline{\tau^{-1}\chi^{-t}\chi} &= \underline{\chi^{-t}\chi\tau^{-1}} \\
\underline{\kappa\chi^t\chi^{-1}} &= \underline{\chi^t\chi^{-1}\kappa} \\
\underline{\kappa^{-1}\chi^{-t}\chi} &= \underline{\chi^{-t}\chi\kappa^{-1}}
\end{aligned}$$

It is easy to see that R contains finitely many relations. The number of relations and the number of quantified relations are independent of x. The length of each relation except for $R6$ is independent of x. For the quantified relations in $R6$, the only thing that depends on x is the value of $q(|x|)$. Anything else can be symbolically written down as the way it is in our coding system (*i.e.*, without further evaluations). The length of each relation is therefore $O(\log|x|)$. (We can also see that each relation (without quantifier) specified in $R6$ has length $O(q(|x|)\log|x|)$ in our coding system.)

We will now show that M' halts on input z iff $(\underline{s}z\underline{\$}\diamond\underline{\tau})\underline{\kappa} \overset{k}{\longleftrightarrow} \underline{\kappa}(\underline{s}z\underline{\$}\diamond\underline{\tau})$ in G with $k \leq$ a fixed polynomial.

Lemma 5.10 *There is a polynomial p such that if $\underline{s}z\underline{\$} \overset{k}{\longleftrightarrow} \underline{h}$ in Γ with $k \leq q(|x|)$ then $(\underline{s}z\underline{\$}\diamond\underline{\tau})\underline{\kappa} \overset{m}{\longleftrightarrow} \underline{\kappa}(\underline{s}z\underline{\$}\diamond\underline{\tau})$ in G with $m \leq p(|x|)$.*

Proof. Assume that $\underline{s}z\underline{\$} \overset{k}{\longleftrightarrow} \underline{h}$ in Γ with $k \leq q(|x|)$. Then

$$\underline{s}z\underline{\$} \equiv W_1 \leftrightarrow W_2 \leftrightarrow \ldots \leftrightarrow W_k \equiv \underline{h},$$

and each $|W_i|$ is bounded by $cq(|x|)\log|x|$ for a fixed constant c. Suppose $W_i \leftrightarrow W_{i+1}$ by applying a rewriting rule $\underline{EqF} = \underline{HpK}$, then either $W_i \equiv \underline{UEqFV}$ and $W_{i+1} \equiv \underline{UHpKV}$, or $W_{i+1} \equiv \underline{UEqFV}$ and $W_i \equiv \underline{UHpKV}$, where U and V are positive words on S_1.

Let X be of the form $\underline{UqV}$, where U and V are words on S_1 and $q \in Q_1$. Then write X^* to denote $\underline{\overline{U}qV}$. Now in $[A;R]$, using one relation in $R3$ and two relations specified in $R6(1)$, we have

$$\begin{aligned}
\underline{\overline{U}(\underline{\overline{E}qF})\underline{V}} &\overset{2}{\longleftrightarrow} \underline{\overline{U}r_i(r_i^{-1}\overline{E}qFr_i)r_i^{-1}V} \\
&\leftrightarrow \underline{\overline{U}(r_i\overline{H}pKr_i^{-1})\underline{V}} \\
&\overset{2}{\longleftrightarrow} \Phi_i(\underline{\overline{UH}pKV})\Psi_i,
\end{aligned}$$

where $\Phi_i = \underline{\chi^{2^{|U|}}\chi^{-1}r_i\chi^{2^{|U|}}\chi^{-1}}$ and $\Psi_i = \underline{\chi^{-2^{|V|}}\chi r_i^{-1}\chi^{-2^{|V|}}\chi}$. It is easy to see that $|\Phi_i| \leq O(\log|x|)$ and $|\Psi_i| \leq O(\log|x|)$ in the way the power strings are coded. So we have

$$W_i^* \overset{3}{\longleftrightarrow} \Phi_i W_{i+1}^* \Psi_i.$$

One can similarly obtain the same result if $W_i \equiv \underline{UHpKV}$ and $W_{i+1} \equiv \underline{UEqFV}$, where $\Phi_i = \underline{\chi^{-2^{|U|}}\chi r_i^{-1}\chi^{-2^{|U|}}\chi}$ and $\Psi_i = \underline{\chi^{2^{|V|}}\chi^{-1}r_i\chi^{2^{|V|}}\chi^{-1}}$.

So we have

$$\underline{s}z\underline{\$} \equiv W_1 \equiv W_1^* \overset{O(k)}{\longleftrightarrow} \Phi W_k^* \Psi \equiv \Phi\underline{h}\Psi \text{ in } G,$$

where $\Phi = \Phi_1\Phi_2\cdots\Phi_{k-1}$, $\Psi = \Psi_{k-1}\cdots\Psi_2\Psi_1$.

Similarly, we can show that $(\underline{s}z\underline{\$})^{-1} \overset{O(k)}{\longleftrightarrow} \Psi^{-1}\underline{h^{-1}}\Phi^{-1}$ in G.

Using a polynomial number of relations in $R4$ and $R6(2)$ of R, we have

$$\begin{aligned}
(\underline{s}z\underline{\$} \diamond \underline{\tau})\underline{\kappa} &\overset{O(k)}{\longleftrightarrow} \Psi^{-1}\underline{h^{-1}}(\Phi^{-1}\underline{\tau}\Phi)\underline{h}(\Psi\underline{\kappa}) \\
&\overset{O(k)}{\longleftrightarrow} \Psi^{-1}[(\underline{h^{-1}\tau h})\underline{\kappa}]\Psi \\
&\leftrightarrow \Psi^{-1}[\underline{\kappa}(\underline{h^{-1}\tau h})]\Psi \\
&\overset{O(k)}{\longleftrightarrow} \underline{\kappa}\Psi^{-1}\underline{h^{-1}}\Phi^{-1}\underline{\tau}\Phi\underline{h}\Psi \\
&\overset{O(k)}{\longleftrightarrow} \underline{\kappa}(\underline{s}z\underline{\$} \diamond \underline{\tau})
\end{aligned}$$

So $(\underline{s}z\underline{\$} \diamond \underline{\tau})\underline{\kappa} \overset{m}{\longleftrightarrow} \underline{\kappa}(\underline{s}z\underline{\$} \diamond \underline{\tau})$ in G for $m \leq p(|x|)$, where p is some fixed polynomial. This completes the proof. ∎

From Lemmas 5.8, 5.9, and 5.10, we have

Lemma 5.11 *If M accepts x in polynomial time, then $(\underline{s}z\underline{\$}\diamond\underline{\tau})\underline{\kappa} \overset{k}{\longleftrightarrow} \underline{\kappa}(\underline{s}z\underline{\$}\diamond\underline{\tau})$ in G for $k \leq p(|x|)$.*

We use Higman-Neumann-Neumann (HNN) extensions to prove the other direction. We first point out that the relations specified in $R6$ can be derived from other relations in R using elementary Tietze transformations.

Lemma 5.12 *Words derived using relations specified in $R6$ can be derived using other relations in R.*

Proof. We need only to demonstrate how to obtain $\underline{r_iV} \overset{*}{\longleftrightarrow} \underline{V\chi^{2^{|V|}}\chi^{-1}r_i\chi^{2^{|V|}}\chi^{-1}}$ using relations not in $R6$, where V is a positive word. The others can be similarly obtained. We prove it by induction on $|V|$. The case of $|V| = 0$ is obvious. For the general case, let $V = V'a$, where $a \in S_1$, then $\underline{r_iV} \overset{*}{\longleftrightarrow} \underline{V'\chi^{k'}\chi^{-1}r_i\chi^{k'}\chi^{-1}a}$, where $k' = 2^{|V'|}$. Keep using relations in $R1$ and $R2$ we have $\underline{V'\chi^{k'}\chi^{-1}r_i\chi^{k'}\chi^{-1}a} = \underline{V'\chi^{k'-1}r_i\chi^{k'-2}\chi a} \leftrightarrow \underline{V'\chi^{k'}r_i\chi^{k'-1}a\chi^2} \overset{*}{\longleftrightarrow} \underline{V\chi^{2k'-1}r_i\chi^{2k'-1}} = \underline{V\chi^{2k'}\chi^{-1}r_i\chi^{2k'}\chi^{-1}}$. This completes the proof. ∎

Hence, two words being equivalent in group G using relations specified in $R6$ for polynomially many times implies that these two words are equivalent in group G without relations specified in $R6$, but they may require using exponentially many other relations. What we are interested here is that as long as $(\underline{s}z\underline{\$} \diamond \underline{\tau})\underline{\kappa}$ is equivalent to $\underline{\kappa}(\underline{s}z\underline{\$} \diamond \underline{\tau})$ in G, without using the redundant relations in $R6$, we will have $\underline{s}z\underline{\$} \overset{*}{\longleftrightarrow} \underline{h}$ in Γ. Then using Lemmas 5.8 and 5.9 we will get what we want. By the way we built Γ in which every relation is in

the form $EqF = HpK$, where p, q are states in Q_1, and E, F, H, K are strings on S_1, the proof given in Rotman [Rot88] can be used in exactly the same way to prove our result here. In particular, our proof is simpler as we need only to consider a special case. We outline the proof below and the reader is referred to [Rot88] for more details.

We first show that the group G is the end result of a chain of HNN extensions. [5]

Let S be a set. Write $\underline{S}$ for $\{\underline{a} : a \in S\}$. Let $\Upsilon = \{r_i : i = 1, ..., |\Gamma|\}$. Define groups G_0, G_1, G_2, and G_3 as follows:

$$\begin{aligned} G_0 &= [\underline{\chi}], \text{ the free group of } \underline{\chi} \\ G_1 &= [\underline{\chi}, \underline{S_1}; R1] \\ G_2 &= [\underline{\chi}, \underline{S_1}, \underline{Q_1}, \underline{\Upsilon}; R1, R2, R3] \\ G_3 &= [\underline{\chi}, \underline{S_1}, \underline{Q_1}, \underline{\Upsilon}, \underline{\tau}; R1, R2, R3, R4] \end{aligned}$$

Lemma 5.13 *1. G is an HNN extension of G_3 with stable letter $\underline{\kappa}$.*

2. G_3 is an HNN extension of G_2 with stable letter $\underline{\tau}$.

3. G_2 is an HNN extension of the free product $G_1 * [\underline{Q_1}]$ with stable letters $\underline{\Upsilon}$.

4. G_1 is an HNN extension of G_0 with stable letters $\underline{S_1}$.

Lemma 5.14 *If $(\underline{s}z\underline{\$} \diamond \underline{\tau})\underline{\kappa} \overset{*}{\longleftrightarrow} \underline{\kappa}(\underline{s}z\underline{\$} \diamond \underline{\tau})$ in G, then there are words V_1 and V_2 on $\underline{\chi}$ and various $\underline{r_i}$ such that $V_1\underline{s}z\underline{\$}V_2 \overset{*}{\longleftrightarrow} \underline{h}$ in G_2.*

Clearly, we can assume that both V_1 and V_2 are reduced words. A word is a-reduced if it contains no sub-words of the form aa^{-1} or $a^{-1}a$.

Lemma 5.15 *Let V_1 and V_2 be words on various $\underline{r_i}$ and $\underline{\chi}$ that are $\underline{r_i}$-reduced for each $\underline{r_i}$. Suppose $V_1\underline{s}z\underline{\$}V_2 \overset{*}{\longleftrightarrow} \underline{h}$ in G_2, then $\underline{s}z\underline{\$} \overset{*}{\longleftrightarrow} \underline{h}$ in Γ.*

By combining all these lemmas we finally get

Lemma 5.16 *M accepts x in polynomial time iff M' halts on z in polynomial time iff $\underline{s}z\underline{\$} \overset{k}{\longleftrightarrow} \underline{h}$ in Γ for $k \leq q(|x|)$ iff $(\underline{s}z\underline{\$} \diamond \underline{\tau})\underline{\kappa} \overset{m}{\longleftrightarrow} \underline{\kappa}(\underline{s}z\underline{\$} \diamond \underline{\tau})$ in G for $m \leq p(|x|)$.*

Now we are ready to finish the proof of Theorem 5.7. We define a reduction f as follows:

$$f(x) = (G, \underline{s}z\underline{\$}, \underline{\tau}, \underline{\kappa}, p(|x|)).$$

Clearly, f is one-to-one. As shown earlier, A contains constant number of symbols depending on M and R contains constant number of relations (some with quantifiers). The length of each coded symbol and the length of each of these relations are $O(\log |x|)$. So the probability distribution of $f(x)$ is proportional to $\frac{1}{P(|x|)} 2^{-|\sigma(x)|}$ for a polynomial P which dominates $\mu(x)$. This completes the proof of Theorem 5.7. ∎

[5] An HNN extension E^* of a finitely presented group $[A; R]$ with stable letters $\{t_i : i \in I\}$ is $E^* = [A, t_i, i \in I; R, t_i^{-1} a_{ij} t_i = b_{ij}, i \in I, j \in J]$ for words a_{ij} and b_{ij} on A such that for each $i \in I$, there is an isomorphism $\phi_i : A_i \to B_i$ with $\phi_i(a_{ij}) = b_{ij}$ for every $j \in J$, where, $A_i = \langle a_{ij} : j \in J \rangle$ (the subgroup generated by a_{ij}'s), and $B_i = \langle b_{ij} : j \in J \rangle$.

6 Completeness Proofs (Part III)

As mentioned earlier in Section 2 that distributional problems under flat distributions cannot be complete under (deterministic) p-time reductions if nondeterministic exponential time is strictly more powerful than deterministic exponential time [Gur91], and it cannot be complete, without any assumption, under one-one, p-length-preserving, and p-time computable reductions [WB95]. Note that the reductions constructed in the completeness proofs above are all one-one and p-length-preserving. Since instances under flat distributions have uniformly small weights on instance size, searching for frequent enough instances of the target problem of a reduction to uniformly dominate frequent instances of the source problem is crucial. Venkatesan and Levin [VL88] suggested to provide reductions with a good random source to help produce instances of the target problem with large enough probability. They showed that the distributional graph edge coloring problem is complete under a randomized reduction by allowing algorithms to toss coins to determine the next moves. Recall that the distribution of this problem is flat. The theory of randomized reductions was further studied by Blass and Gurevich [BG91, BG93].

6.1 Randomized Reductions

We assume that a randomized algorithm does not flip a coin unless the computation requires a random bit. For simplicity, coins are assumed to be unbiased. Randomized algorithms (to solve a problem) are allowed to make errors and produce incorrect outputs on some sequences of random bits. They could also run forever on some other random (infinite) sequences. Let $\mathcal{A}$ be a randomized algorithm. If $\mathcal{A}$ on input x halts and produces a correct output with s being the random sequence it generates, then (x, s) is called a *good* input for $\mathcal{A}$. Clearly, $\mathcal{A}$ runs deterministically on (x, s). If $\mathcal{A}$ on input x runs deterministically, then (x, e) is a good input, where e denotes the empty string. Let Γ be a set of good inputs for $\mathcal{A}$ and $\Gamma(x) = \{s : (x, s) \in \Gamma\}$. Let μ be an input distribution. If for all x with $\mu(x) > 0$, $\Gamma(x)$ is non-empty, we call Γ a *good-input domain* of $\mathcal{A}$ (with respect to μ). Clearly, no string in $\Gamma(x)$ is a prefix of a different string in $\Gamma(x)$, for, otherwise, the longer string cannot be in $\Gamma(x)$ as the algorithm stops before it can be generated. Let $U_\Gamma(x) = 1/\sum_{s\in\Gamma(x)} 2^{-|s|}$, which is called the *rarity* function of Γ [BG93]. The smaller the value of the rarity function, the more good random sequences there are for the algorithm. If for all x, $U_\Gamma(x) = 1$, the randomized algorithm produces a correct output with probability 1. For our purpose, we only need to require that the value of $U_\Gamma(x)$ be "reasonable" in the sense that U_Γ is polynomial on average.

Definition 6.1 Let $\mathcal{A}$ be a randomized algorithm and μ an input distribution.

1. Let Γ be a good-input domain of $\mathcal{A}$. Then Γ is *nonrare* (with respect to μ) if the rarity function U_Γ is polynomial on μ-average. Define μ_Γ by, for all $(x, s) \in \Gamma$, $\mu_\Gamma(x, s) = \mu(x)2^{-|s|}U_\Gamma(x)$.
2. $\mathcal{A}$ runs in time *polynomial on μ-average* if there exists a nonrare good-input domain Γ such that $\sum_{(x,s)\in\Gamma} t^\varepsilon(x, s)|x|^{-1}\mu_\Gamma(x, s) < \infty$ for some $\varepsilon >$

0, where $t(x,s)$ is the running time of $\mathcal{A}$ on input x with random sequence s. $\mathcal{A}$ runs in *polynomial time* if $t(x,s)$ is bounded by a polynomial in $|x|$ for all $(x,s) \in \Gamma$.

3. $\mathcal{A}$ is *almost total* if for all x with $\mu(x) > 0$, $U_\Gamma(x) = 1$.

For decision problems, if the randomized algorithm also provides a witness along with a yes/no answer, then the correctness of the answer can be verified by evaluating the witness (like search problems). If a witness is not provided, then one way to justify the correctness of the output is to make sure that the input is good. For example, we may need to require that the good-input domain Γ (with respect to μ) be *certifiable* [BG93], meaning that for every input (x,s), whether $(x,s) \in \Gamma$ is deterministically decidable in ap-time with respect to distribution $\mu(x)|s|^{-2}2^{-|s|}$.

By Definition 6.1, an ap-time randomized algorithm on input x produces a correct output with probability $1/U_\Gamma(x)$, which could be small. While this may not seem satisfactory, Blass and Gurevich [BG93] showed that by iterating such an algorithm in an appropriate manner, a correct solution will be produced with probability 1 in average polynomial time. We assume that there is an efficient way to check whether an output of an iteration is correct as discussed above. Since a randomized algorithm $\mathcal{A}$ may run a much longer time on inputs not in the good-input domain and it may not even halt on some bad inputs, a new iteration should start early without waiting for the previous one to stop. One way to carry it out is to use the "round-robin" method. Denote by $\mathcal{A}^*$ the iteration of $\mathcal{A}$. At stage n of $\mathcal{A}^*$, it independently carries out one step in the first, second, ..., and the n-th iterations of $\mathcal{A}$ respectively in that order. $\mathcal{A}^*$ stops as soon as one of the iteration stops with a correct output. So $\mathcal{A}^*$ is a randomized algorithm whose sequence of random bits on input x is the combination of random bits of each iteration in the round-robin fashion on the same input. This is equivalent to saying that $\mathcal{A}^*$ flips a coin only when an iteration asks for one and passes the random bit to that iteration. Several other iteration schemes have also been studied in [BG93, WB93b].

Theorem 6.1 *Let $\mathcal{A}$ be a randomized algorithm with input distribution μ. Then $\mathcal{A}$ runs in time polynomial on μ-average iff $\mathcal{A}^*$ runs in time polynomial on μ-average and is almost total.*

Definition 6.2 Let (A,μ) and (B,ν) be distributional decision problems. Then (A,μ) is *ap-time randomly reducible to* (B,ν) if there is a reduction f, computable by a randomized algorithm in time polynomial on μ-average with a nonrare good-input domain Γ such that the following codistions are satisfied.

1. The correctness of the output can be efficiently verified.
2. For all $(x,s) \in \Gamma$, $x \in A$ iff $f(x,s) \in B$.
3. μ_Γ is weakly dominated by ν with respect to f.

If the reductions f can be computed by p-time randomized algorithms in both cases, then the reductions are called *p-time randomized reductions*.

A distributional decision problem (D, μ) is solvable in *randomized ap-time* if D is decidable by a randomized algorithm in time polynomial on μ-average with a nonrare good-input, and the correctness of the output of the algorithm can be efficiently verified. Denote by RAP the class of all such problems.

The randomized reductions defined in Definition 6.2 are closed for randomized ap-time and they are transitive.

Lemma 6.2 *(1) If (A, μ) is ap-time randomly reducible to (B, ν) and (B, ν) is in* RAP*, then so is (A, μ). (2) The ap-time randomized reductions are transitive.*

6.2 Distributional Halting (Version 2)

Using a randomized reduction, it is straightforward to show that the distributional halting problem (version 2) is complete for DistNP although it cannot be complete under a one-one and length preserving p-time reduction. Denote by K_2 the set of positive instances (i, x, t) of the distributional halting problem (version 2). The probability distribution $\mu_{K_2}(i, x, t)$ of instance (i, x, t) (positive or negative) is proportional to $2^{-(l+m+n)}l^{-2}m^{-2}n^{-2}$, where $l = |i|$, $m = |x|$, and $n = |t|$. Recall that the size of (i, x, t) is $|i| + |x| + |t|$ while the size of instance (i, x, n) of distributional halting problem (version 1) is $|i| + |x| + n$. Clearly, K_2 is NP-complete and μ_{K_2} is flat.

Theorem 6.3 *The distributional halting problem (version 2) is complete for* DistNP.

Proof. We reduce (K_1, μ_1) to (K_2, μ_2) by a p-time randomized reduction as follows. On instance (i, x, n) of K_1, the reduction flips a coin n times to produce a random string s, and then outputs (i, x, s). More precisely, we define a good-input domain Γ by $\Gamma = \{(y, s) : y = (i, x, n) \text{ and } |s| = n\}$. Clearly, the rarity function $U_\Gamma(x) = 1$ and Γ is p-time computable and so is certifiable. Let $\sigma_i((x_1, x_2, x_3)) = x_i$ for $i = 1, 2, 3$. For all $(y, s) \in \Gamma$, let $f(y, s) = (\sigma_1(y), \sigma_2(y), s)$. Then f is one-one and p-time computable. Clearly, for all $(y, s) \in \Gamma$: $y \in K_1$ iff $f(y, s) \in K_2$. To check the domination property, we note that $\mu_\Gamma(y, s) = \mu_{K_1}(y)2^{-|s|} = \mu_{K_2}(f(y, s))$. Thus, f is the desired p-time randomized reduction. ∎

6.3 Distributional Graph Edge Coloring

We deal with directed graphs throughout this section, where self-loops are allowed and nodes are labeled. For simplicity, we simply use graphs to denote such directed graphs. Graphs are sampled by first selecting the number of nodes n at random with respect to the default uniform distribution, then selecting a graph of n nodes at random with uniform distribution (*i.e.*, all graphs of n nodes have the same probability).

For convenience, we will consider the distributional tiling problem in which corners (instead of edges) of tiles are marked with letters.

Venkatesan and Levin [VL88] constructed a randomized reduction from the distributional tiling problem to the distributional graph edge coloring problem.

The proof presented here contains a number of modifications, and we provide all the details to make the proof accessible. Some of these details are extracted from [Ven91].

Let G be a graph. We also use G to denote the set of nodes in G. Denote by $\mathbf{E}(X)$ the expected value of the random variable X. Write $f(n) \sim g(n)$ if $\lim_{n\to\infty} f(n)/g(n) = 1$. If the symbols o or $\sim$ are used without a variable, then we mean that the relation holds as $n \to \infty$. Let the graph G be chosen at random. A property ϕ of G is satisfied almost everywhere (or for almost every (*a.e.*) graph) if ϕ occurs in G with probability $1 - o(1)$.

The following two lemmas on probabilities and approximations are useful tools in studying random graphs. Their proofs are standard and can be found from numerous sources such as [Bol85].

Lemma 6.4

$$n! \sim \sqrt{2\pi n}\left(\frac{n}{e}\right)^n \quad \textit{(Stirling's approximation)} \tag{6.1}$$

$$\binom{n}{k} \sim \begin{cases} n^k e^{-\frac{k^2}{2n} - \frac{k^3}{6n^2}}, & \textit{if } k = o(n^{3/4}) \\ n^k, & \textit{if } k = o(\sqrt{n}) \end{cases} \tag{6.2}$$

$$\frac{\binom{n-k}{m-k}}{\binom{n}{m}} \sim \left(\frac{m}{n}\right)^k, \ \textit{if } m \le n \textit{ and } k = o(\sqrt{m}) \tag{6.3}$$

Let $0 < \lambda < 1$ and λn be an integer. Let $H(x) = -x\log x - (1-x)\log(1-x)$, which is called the binary entropy function. Let $J(\lambda, n) = \sqrt{2\lambda(1-\lambda)\pi n}$. Then

$$\frac{2^{nH(\lambda)}}{2J(\lambda,n)} \le \binom{n}{\lambda n} \le \frac{2^{nH(\lambda)}}{J(\lambda,n)} \tag{6.4}$$

Note that $H(x) = H(1-x)$ in the interval $[0,1]$, $H(1) = H(0) = 0$, and $H(1/2) = 1$ is the maximum.

Lemma 6.5 *Let X be the number of successes in n independent Bernoulli trials with individual probability of success $p = o(1)$, $np = n^{2\varepsilon}$. Then the following results can be obtained by using the above approximations and the central limit theorem.*

1. $\mathbf{E}(X) = np \sim np(1-p) = \mathbf{Var}(X)$.
2. *With probability $1 - o(1)$, $X = 0$ if $\varepsilon < 0$, and $X \ge n^{2\varepsilon} - n^{\varepsilon}\log n$ if $\varepsilon > 0$.*
3. $\mathbf{Pr}[X = np] \ge n^{-\varepsilon}/6$. *Namely,* $\mathbf{Pr}[X = np] \ge \frac{1}{6\sqrt{np}}$.

Certain graph structures are needed for constructing the reduction. We will show that such structures exist on polynomial fraction of graphs. We first consider *transitive tournaments*, where a transitive tournament (or simply *tournament*) is an acyclic (expcept for self-loops at each node) complete digraph.

Lemma 6.6 *Let c be a non-negative real number. Let X_k be the number of k-node tournaments in a random graph of n nodes, where $n = 2^{k-c} > k^3$. Let $m = 2^k$. Then $\mathbf{E}(X_k) \sim m^{-c}$ and $\mathbf{E}(X_k^2) \sim m^{-c} + m^{-2c}$.*

Proof. Let $\mathcal{K}$ be the set of all k-node subgraphs and for $A \in \mathcal{K}$, let $I(A) = 1$ if A is a tournament and 0 otherwise. Then $X_k = \sum_{A \in \mathcal{K}} I(A)$. Let N_l be the random variable for the number of pairs of k-node tournaments sharing $k-l$ nodes. Then $\mathbf{E}(X_k^2) = \mathbf{E}(\sum_{A,B \in \mathcal{K}} I(A)I(B)) = \sum_l \mathbf{E}(N_l)$. To calculate $\mathbf{E}(N_l)$, we first calculate the number of possible pairs of k-node tournaments from n-node graphs that share $k-l$ nodes. It is easy to see that for m labeled nodes there are exactly $m!$ tournaments. So the number of possible pairs is

$$\binom{n}{k} k! \binom{n-k}{l} \binom{k}{k-l} \frac{k!}{(k-l)!} = \binom{n}{k} \binom{n-k}{l} k! l! \binom{k}{k-l} \binom{k}{l}.$$

Since for each pair of nodes, there are four possible ways to form an edge (including the empty edge), the probability of each pair is

$$4^{-\left[2\binom{k}{2} - \binom{k-l}{2}\right]} 2^{-(k+l)} = 2^{-(k^2-l^2+2kl)}.$$

It follows that $\mathbf{E}(N_l) = 2^{-(k^2-l^2+2kl)} \binom{k}{l}^2 n!/(n-k-l)!$. Note that $k+l = o(\sqrt{n})$. Using Stirling's approximation, we get

$$\begin{aligned}
n!/(n-k-l)! &\sim \frac{\sqrt{2\pi n}(n/e)^n}{\sqrt{2\pi(n-k-l)}((n-k-l)/e)^{n-k-l}} \\
&\sim \left[\left(\frac{n}{n-(k+l)}\right)^{n/(k+l)}\right]^{(k+l)} \left(\frac{n-(k+l)}{e}\right)^{k+l} \\
&\sim e^{k+l}(n/e)^{k+l} \\
&\sim n^{k+l}.
\end{aligned}$$

Hence,

$$\mathbf{E}(N_l) \sim n^{k+l} \binom{k}{l}^2 2^{l^2-k^2-2kl+k+l} \tag{6.5}$$

$$= \binom{k}{l}^2 2^{-c(k+l)} 2^{l(l-k)} \tag{6.6}$$

Putting $m = k$, we get $\mathbf{E}(N_0) \sim m^{-c}$ and $\mathbf{E}(N_k) \sim m^{-2c}$.

Denote by $\Phi(l)$ the expression (6.6) above. Then $\mathbf{E}(X_k^2) = \sum_{l=0}^{k} \mathbf{E}(N_l) \sim m^{-c} + m^{-2c} + \sum_{l=1}^{k-1} \Phi(l)$. We will show that for $0 < l < k$, either $k\Phi(l) = o(m^{-c})$ or $k\Phi(l) = o(m^{-2c})$.

It follows from (6.6) that $\frac{\Phi(l+1)}{\Phi(l)} = \left(\frac{k-l}{l+1}\right)^2 2^{-c} 2^{2l-k+1}$. Note that $\frac{1}{k} < \frac{k-l}{l+1} < k$. So for sufficiently large k, we get

$$\frac{\Phi(l+1)}{\Phi(l)} < k^2 2^{-c+1} 2^{-k/3} < 2^{-k/4}, \text{ if } 0 \le l \le \tfrac{k}{3} \tag{6.7}$$

$$\frac{\Phi(l+1)}{\Phi(l)} \ge \frac{2^{-c+1}2^{k/3}}{k^2} > 2^{k/4}, \text{ if } \tfrac{2k}{3} \le l \le k \tag{6.8}$$

It follows from (6.7) that in the interval $[0, \frac{k}{3}]$, $\Phi(l)$ decreases monotonically starting from $\Phi(0) = m^{-c}$ and that $\Phi(l)/m^{-c} < 2^{-k/4} = o(1/k)$. Similarly from (6.8), $\Phi(l)$ increases monotonically and reaches the maximum value at $\Phi(k) = m^{-2c}$ and so $m^{-2c}/\Phi(l) > 2^{k/4}$. This implies that $\Phi(l)/m^{-2c} \leq 2^{-k/4} = o(1/k)$. Let $\lambda = l/k$. When $\lambda \in [\frac{1}{3}, \frac{2}{3}]$, using (6.4) and (6.6), we get

$$\Phi(l) < \frac{2^{2kH(\lambda)}}{2\lambda(1-\lambda)\pi k 2^{-c(1+\lambda)k} 2^{-\lambda(1-\lambda)k^2}} < 2^{-\Theta(k^2)}.$$

Thus,

$$\begin{aligned}
\mathbf{E}(N_l) &\sim m^{-c} + m^{-2c} + m^{-c} \sum_{0<l<\frac{k}{3}} \frac{o(1)}{k} \\
&\quad + \sum_{\frac{k}{3} \leq l \leq \frac{2k}{3}} 2^{-\Theta(k^2)} + m^{-2c} \sum_{\frac{2k}{3}<l<k} \frac{o(1)}{k} \\
&\sim m^{-c} + m^{-2c}.
\end{aligned}$$

∎

Corollary 6.7 *(1) If $c > 0$, then almost every graph has no k-node tournament. (2) A random graph on $n = 2^k$ nodes has a unique largest $(k+l)$-node tournament with probability at least $\Omega(n^{-2l})$.*

Proof. The first statement follows from $\mathbf{Pr}[X_k \geq 1] \leq \mathbf{E}(X_k) = m^{-c} = o(1)$. This bounds the size of the largest tournament. To see the second statement, note that $c = 0$ implies $\mathbf{E}(X_k) = m^{-c} \sim 1$, and so a k-node tournament exists in almost every graph. The probability that a node extends a given k-node tournament into a $(k+1)$-node tournament is $\frac{1}{n}\left(\frac{3}{4}\right)^k > 2^{-2k} = 1/n^2$. Thus, the probability of a graph with a unique largest $(k+l)$-node tournament is $\Omega(n^{-2l})$. ∎

Remark 6.1 The second statement of Corollary 6.7 is not true for *a.e.* graph. Here we deal with some properties which are not true for *a.e.* graph but hold with only a polynomial probability. ∎

The following lemma is used to, on a given tournament, generate a random graph such that the given tournament is the largest and unique. Note that a complete directed graph (and so a tournament) always contains a Hamiltonian path that can be found in time polynomial of the number of nodes (see, for example, [Liu68]).

Lemma 6.8 *Let G be a graph on $\{1, \ldots, 2^{k-1}+k\}$ with a tournament on nodes $1, 2, \ldots, k$ as the Hamiltonian path and all edges $\langle i, j\rangle$ ($i > k$ or $j > k$) chosen at random. Then with probability $1 - o(1)$, G has no other tournament of size k.*

Proof. By Corollary 6.7 (1), no k-node tournaments exist on $\{k+1, \ldots, 2^{k-1}+k\}$ in almost every graph G. However, an l-node tournament disjoint with $\{1, \ldots, k\}$ may share $k-l$ nodes and form a k-node tournament. We are to show that the expected number n_l of such tournaments is $o(1)$. Note that n_l is the same as $E(N_l)$ in Lemma 6.6 except that the nodes and edges of the k-node tournament on $\{1, \ldots, k\}$ are fixed here. Hence, by (6.2) and Stirling's approximation,

$$\begin{aligned} n_l &= \mathbf{E}(N_l)\frac{4^{\binom{k}{2}}}{k!\binom{2^{k-1}+k}{k}} \sim \mathbf{E}(N_l)\frac{2^{k^2}}{2^{(k-1)k}2^{k\log k + o(k\log k)}} \\ &< 2^{-k} = o(1)/k. \end{aligned}$$

Summing over all possible values of l, the result follows. ∎

A degree of a node is the number of (incoming and outgoing) edges connecting to the node. Write $d(G)$ for maximal degree of nodes in G. An *embedding* is a one-one homomorphism. Note that in our sampling model, edges are chosen independently.

Lemma 6.9 (**Embedding Lemma**) *Let H be an induced subgraph of F. Let $n = |F| - |H|$ and $d(F-H) = o(n)$. If $\varepsilon \cdot d(F-H) < 1 - 2\log\log n/\log n$ and the probability of choosing an edge is at least $n^{-\varepsilon}$, then F can be embedded into a.e. graph G that contains H as an induced subgraph and $|G| = |F|$. Moreover, an embedding g can be found in polynomial time such that g is identical on H.*

Proof. Partition $F-H$ into independent sets I_j, $j \leq d+1$, where $d = d(F-H)$, such that $|I_j| \geq n/(d+1)$. Split $G-H$ into disjoint sets V_j, $|V_j| = |I_j|$. Initially, let $g(u) = u$ for all $u \in H$. Repeat the following step until g is extended as required.

Extension step. Let V be the nodes on which g is defined (*i.e.*, embedding has been found). Call $v \in V_j$ a candidate for $u \in I_j$ if letting $g(u) = v$ would extend g as an embedding over $V \cup \{u\}$. For this, v must satisfy at most d many connectivity constraints. Consider the bipartite graph formed by connecting every node in I_j to all of its candidates in V_j. This graph has the edge probability $O(n^{-\varepsilon d})$, and by the condition on ε, *a.e.* such bipartite graph has a perfect matching (see [Bol85]). This matching finds a node satisfying the connectivity requirements for every node in I_j and extends g over $H \cup I_j$.

The extensions are independent as I_j are disjoint and the failure probability of each extension is exponentially small. ∎

Write $u \to v$ if there is an outgoing edge from u to v. Let $t_1 \to t_2 \to \ldots \to t_k$ be a Hamiltonian path of a tournament T, where $k = |T|$. The code of a node u is a binary string of length $2|T|$ in the form $c(t_1, u)c(u, t_1) \cdots c(t_k, u)c(u, t_k)$, where $c(u, v) = 1$ if $u \to v$ and 0 otherwise [BES80]. The code of u with respect to $t_i \in T$ is the restriction of its code to the digits $2i-1$ and $2i$.

A looped node is called *blank* if its self-loop is not colored.

Theorem 6.10 *The distributional graph edge coloring problem is complete for* DistNP.

Proof. We fix a reduction from (K_1, μ_{K_1}) to the distributional tiling problem. Let $X = (\mathcal{T}, m, S)$ ($|S| < m$) be an instance of the distributional tiling problem produced by the reduction, where $\mathcal{T}$ is a set of constant number of tiles with marked corners. We distinguish the tiles by coloring the corners of the tiles. Without loss of generality, we assume that the following particular tiles are included. Namely, the lower left corner is colored blank, which represents the initial state of a universal Turing machine with the head looking at the right, and other three corners are colored with non-blank colors (one such tile encodes the case when the first bit of the input to the universal Turing machine is 0, and the other encodes the case when the first bit of the input is 1). Note that tiles cannot be rotated or turned over. Any other tiles will have all their corners colored by non-blank colors. The first tile in S has a blank lower left corner, and S encodes the input of K_1 (for example, if the ith digit of the input is 1, then the lower right corner of the ith tile of S is marked with 1). Clearly, The number of colors depends only on a fixed universal Turing machine that accepts K_1, and so is a constant. We may use a small universal Turing machine (in terms of the number of states and symbols) to reduce the number of colors. Small universal Turing machines have been studied in [Sha56, Ike58, Wat61, Min67, Pre79] and other literature.

Let

$$\kappa = \log_{3/2}(2m)^2,\ k = \lceil \kappa \rceil,\ n = 2^{k-2}.$$

Note that $n = 2^{\kappa+\delta-2}$ for some $\delta \in [0, 1)$. Hence, $n = 2^{-2+\delta}(2m)^{2/\log 1.5}$, which implies that $2m^{3.4} < n < 8m^{3.45}$.

We partition $\{1, \ldots, 2n + k\}$ at random into disjoint sets

$$T,\ L_T,\ L_1,\ U_T,\ U_1$$

such that

$$|T| = k,\ |L_T \cup L_1| = n,\ |U_T \cup U_1| = n,$$

and

$$|L_T| = m^2 - 1,\ |U_T| = m^2.$$

Let $L = L_T \cup L_1$ and $U = U_T \cup U_1$.

We then randomly order T as $t_1, t_2, \ldots, t_k$ and extend it by randomly adding edges into a graph of $2n + k$ nodes. The graph is random except that the following properties are enforced.

- **P1** T forms a tournament and $t_1 \to t_2 \to \ldots \to t_k$ forms a Hamiltonian path. (It can be shown that T is the largest and unique tournament in a polynomial fraction of graphs, which will be used to enforce the graph structures and the color pattern.)
- **P2** $T \cup L$ is the set of all nodes with self-loops.
- **P3** Every node in $U_T \cup L_T$ is connected to every node in T.

Arrange the nodes in U_T in the decreasing order of codes as $v_1, v_2, \ldots, v_{|U_T|}$ and connect t_k to v_1. [6] Then add or delete the edges between successive input

[6] We will show in Lemma 6.12 that all these codes are distinct in *a.e.* G and so the input nodes on decreasing order can be uniquely identified from the tournament T.

nodes v_i, v_{i+1} to encode S: the edge $v_i \to v_{i+1}$ encodes 0 (meaning that the ith bit of the input is 0), the edge $v_i \leftarrow v_{i+1}$ encodes 1 (meaning that the ith bit of the input is 1), and the absent of edge between v_i and v_{i+1} means there is no input on position i. The nodes in U_T are called *input nodes*. The resulted graph is the desired graph G.

Lemma 6.11 *The probability distribution of $(2n+k)$-node graphs that satisfy all the three properties of* **P1**, **P2**, *and* **P3** *is at least $\Omega(m^{-18})$.*

Proof. We show that a random graph G' of $2n+k$ nodes has the required distribution. The probabilities mentioned below in **p1**, **p2**, and **p3** are the probabilities of satisfying **P1**, **P2**, and **P3**, respectively.

p1 Note that $2n+k = 2^{k-1}+k$. It follows from Corollary 6.7 (2) that a 2^{k-2}-node random graph has a unique largest k-node tournament with probability $\Omega((2n)^{-4}) = \Omega(n^{-4})$. So with probability at least $\Omega(n^{-4})$ a random G' has a k-node tournament. By Corollary 6.8, *a.e.* G' has no k-node tournament other than T.

p2 A random G' has exactly $n+k$ looped nodes with probability

$$\begin{aligned}
\binom{2n+k}{n+k}2^{-(2n+k)} &\sim \binom{2n}{n}\left(\frac{2n+k}{n+k}\right)^k 2^{-(2n+k)} \\
&\quad \text{(By Approximation (6.3))} \\
&\sim \binom{2n}{n}2^k 2^{-(2n+k)} \\
&\geq \frac{2^{H(\frac{1}{2})2n}}{\sqrt{4\pi n}2^{2n}} \quad \text{(By Inequality (6.4))} \\
&= \frac{1}{\sqrt{4\pi n}}
\end{aligned}$$

p3 Note that there are 3^k possible ways for each node not in T to connect to every node in T. Also, $k = \kappa + \delta$, where $0 \leq \delta < 1$. So the probability of success is $p = (3/4)^k$. Hence, the expected number of looped nodes in a graph satisfying **P1** and **P2** and connected to every node in T is

$$|L|\left(\frac{3}{4}\right)^k = 2^{k-2}\left(\frac{3}{4}\right)^k = 2^{-2}\left(\frac{3}{2}\right)^k = m^2(3/2)^\delta.$$

It follows from Lemma 6.5 (3) that

$$\mathbf{Pr}[|L_T| = m^2 - 1] \geq \Omega\left(\frac{1}{m}\right).$$

We now deal with unlooped nodes. Similarly, we can show that the expected number of unlooped nodes in a graph satisfying **P1** and **P2** and connected to every node in T is $m^2(3/2)^\delta$, and so

$$\mathbf{Pr}[|U_T| = m^2] \geq \Omega\left(\frac{1}{m}\right).$$

Thus, all these properties hold with probability

$$\Omega\left(\frac{1}{n^{4.5}}\cdot\frac{1}{m^2}\right) > \Omega\left(\frac{1}{m^{15.75+2}}\right) > \Omega(m^{-18}).$$

∎

It is easy to see, from the above proof, that the number of random bits needed to generate G is $\Theta(n^2) = \Theta(m^{6.9})$. Hence, we may consider random strings of length m^7 starting with $\lceil 18 \log m \rceil$ zeros. The rest of the random bits are sufficient to generate a random graph G with the desired structure. We will show (in Lemma 6.13 below) that for *a.e.* such random string s, S can be extended to tile an m^2 squre using tiles from $\mathcal{T}$ iff G is colorable under the given color specification and constraints (which will be described later). Let $\Gamma(X)$ be the set of all such random strings s. Then $\Gamma = \{(X, s) : s \in \Gamma(x)\}$ is a good-input domain. Let $\Gamma'(X) = \{s : |s| = m^7 \text{ and } s = 0^l s' \text{ for } l = \lceil 18 \log m \rceil\}$. Then $U_{\Gamma'}(X) = m^{-18}$. Since $U_{\Gamma'}(X) = U_{\Gamma}(X) + o(1)$, U_Γ is non-rare.

The desirable reduction $f(X, s)$ (for $(X, s) \in \Gamma$) is the graph G we constructed above plus a color specification and a set of constraints that we will describe later. We first describe the structures we need for coloring. The essential part of the structures is restricted to the nodes in $T \cup L_T \cup U_T$.

Remark 6.2 From now on, unless otherwise stated, by *a.e.* G we mean *a.e.* graph produced by f on Γ'. ∎

Structures

For convenience, we will use various types of *links* to classify edges. Links will be directional (*i.e.* asymmetric), Each type of links are associated with a symbolic name (written in bold face) and a direction. The reduction will contain information about these links. A link between u and v can be identified by inspecting the two-node spot (*i.e.* the induced edge with the induced color). Write $u \xrightarrow{\mathbf{h}} v$ to denote that u is connected to v by an **h**-link. The following types of links will be used frequently: **h** (for horizontal), **v** (for vertical), $\mathbf{d_1}$, $\mathbf{d_2}$ (for diagonal), **T** (for toroidal), **I** (for input chain), and **K** (for key). The meanings of these links will be given below when they are introduced.

If on a subset of nodes, exactly one outgoing and incoming links of the same class (for example, **h**, **v**) exist for every node, then that class naturally induces a permutation on this subset of nodes. We then write, for instance, $v = \mathbf{h}(u)$ and $u = \mathbf{h}^{-1}(v)$ accordingly.

When a self-loop is colored, it is convenient to simply say that the looped node is colored (with the same color of the self-loop).

Remark 6.3 It is essential in the reduction construction that any colored looped node has at most one incoming and/or outgoing link of each type mentioned above. ∎

The Input Chain

Lemma 6.12 *In a.e. G, nodes in U_T have distinct codes.*

Proof. It is straightforward to show the following claim. Let i integers be chosen uniformly and independently at random from $\{1, 2, \ldots, n\}$, where $i = o(\sqrt{n})$. Then the probability that all these numbers are distinct is

$$\prod_{j=1}^{i-1} \left(1 - \frac{j}{n}\right) = e^{-\Theta\left(\frac{i^2}{n}\right)} = 1 - o(1).$$

A code word for a node in U_T is a sequence of k pairs of binary digits, where the i-th pair of digits corresponds to the connection to the i-th node in T. Except for 00, all other three combinations are possible. So there are 3^k possible code words for a node in U_T. By construction,

$$|U_T| = m^2 = \frac{1}{4} \cdot \left(\frac{3}{2}\right)^{\kappa} < 3^{k \log_3 1.5} = o(3^{0.36k}) = o(\sqrt{3^k}).$$

Take $N = 3^k$ and $i = |U_T|$ in the above claim, then the lemma follows. ∎

Recall that nodes in U_T are called input nodes and there is an edge from t_k to v_1. Also, $|U_T| = m^2$. Let $l = m^2$. Thus, in *a.e.* G, a monotonically decreasing codes of the input nodes as $v_1, v_2, \ldots, v_l$ produces a unique sequence. This ordering can be illustrated as follows. For each i, let $l(i) - 1$ be the length of the maximal joint prefix of the codes of v_i and v_{i+1}. The edge connecting v_i and $t_{l(i)}$ is called a **K**-link. A **K**-link connecting v_i to T corresponds to the $l(i)$th digit of its code.

Assign nodes $z_i \in L_T$ to form a path: $t_k \to v_1 \to z_1 \to v_2 \to z_2 \to \ldots \to z_{l-1} \to v_l$ where $z_i \neq z_j$ for $i \neq j$. We call this path an *input chain*. The edges on the path are called **I**-links. There are exactly $|U_T|$ **K**-links and exactly $|U_T| + |L_T|$ **I**-links.

Toroidal Grids

A *product* of two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ is the graph $G = (V, E)$ such that $V = V_1 \times V_2$ and $E = \{\langle (u_1, v_1), (u_2, v_2) \rangle : \langle u_1, u_2 \rangle \in E_1 \text{ or } \langle v_1, v_2 \rangle \in E_2\}$. A *toroidal grid* is a graph obtained by adding at most one of the diagonals to the smallest squares in the product of two directed cycles.

Consider a directed cycle $C_m = 1 \to 2 \to \cdots \to m \to 1$ of order m, where each node has a self-loop. The product of C_m by C_m induces an $m \times m$ toroidal grid pattern called a *primary grid*. Except for the edges induced from $m \to 1$ (called the *toroidal edges*), all edges in the same row point to the same direction, and so do the edges in the same column. Horizontal edges (including the horizontal toroidal edges) in the grid are called **h**-links, and vertical edges (including the vertical toroidal edges) are called **v**-links. (Some toroidal edges are neither horizontal nor vertical. We simply refer them as toroidal edges.) Clearly, $\mathbf{h}^m$ and $\mathbf{v}^m$ are identities. Let $\mathbf{d_1} = \mathbf{hv}$ and $\mathbf{d_2} = \mathbf{h^{-1}v}$, which define the diagonals in the smallest squares of the toroidal grid. It is easy to see that the toroidal grid guarantees the following connectivity: for any node u on the grid, $u = \mathbf{h}^i\mathbf{v}^j(1)$ for some i and j.

We now consider a toroidal grid D by arranging all the nodes in $L_T \cup \{t_k\}$ to pass through the entire primary grid such that the bottom row is, from left to right, $t_k, z_1, z_2, \ldots, z_{m-1}$, and the next row is $z_m, \ldots, z_{2m-1}$, etc.

Hence, each looped node in L_T has at most one link of each of the following types: **T**-link, incoming and outgoing **h**, **v**, $\mathbf{d}_1$, $\mathbf{d}_2$, and **I** links. Each unlooped node in U_T has at most one incoming (respectively, outgoing) **I**-link. t_k has no incoming **I**-link.

We note that in the input chain and the grid pattern we described above, all nodes except t_k have a bounded degree. Let F be a graph that contains T, the toroidal grid D, and the input chain as induced subgraph. Let $H = T$ in the Embedding Lemma. It is easy to extend F to a graph of $2n + k$ nodes such that all nodes in $F - H$ have a bounded degree, and so the condition of the Embedding Lemma is satisfied. Hence, the Embedding Lemma guarantees the existence of the desired structure (the input chain and the grid) in *a.e.* G. The embedding is identical on T and preserves all types of links. By the Embedding Lemma, the embedding can be found in polynomial time. Note that the embedding gives a permutation of the indexing for nodes in L_T and U_T we described above. So without loss of generality, we still assume the same indexing in the discussion below.

Coloring the Random Graph

The following edges are colored blank: all the **I**-links, all the **K**-links, all the **T**-links, all the self-loops in T, all the edges in the fixed Hamiltonian path of T, all the **h**-links, and all the **v**-links. Let R specify these as constraints.

The diagonals $\mathbf{d}_1$-links and $\mathbf{d}_2$-links are distinguished by two different colors. All other edges not specifically mentioned above are colored by a color that has not been used in coloring the tiles in $\mathcal{T}$.

The smallest squares in the toroidal grid H can be viewed as tiles. The initial tiling sequence S of X is determined by the connectivity pattern of the chain $v_1, v_2, \ldots, v_{|S|}$ as described before, which will be represented by coloring the looped nodes in the bottom row of the toroidal grid, exactly the same as the correspondent colors in S.

If X is a positive instance of the distributional tiling, then the tiling pattern can be mapped to the grid. So all the looped nodes in L_T can be colored by embedding a tile in the smallest square of the grid iff X is a positive instance. Clearly, the number of colors used to color the graph is a constant, which depends only on the fixed set of tiles $\mathcal{T}$.

On the toroidal grid H, $(i, m) \to (j, 1)$ and $(m, i) \to (1, j)$, $1 \leq i, j \leq m$, are the toroidal edges and the edge $(1, m) \to (m, 1)$ is a double edge. So there are $2m^2 - 1$ toroidal edges. There are $2m(m + 1)$ **h** and **v**-links that are non-toroidal edges. On the input chain, there are $2m^2 - 1$ **I**-links and m^2 **K**-links. On the tournament T, there are k self-loops and $k-1$ edges on the Hamiltonian path. Thus, the number of blank edges in a colored graph that exhibits the tiling pattern is equal to

$$b = 7m^2 + 2m + 2k - 3.$$

The coloration is (C, b), where C is the set of all possible spots induced from colored graphs that represent tiling patterns for positive instances X. It is not difficult to describe these spots. For instance, C includes all spots

induced from colored tiles (by mapping a colored tile into one of the smallest square of the grid), but C does not contain spots that contain a looped loop colored by the default color while all edges are colored blank. Clearly, if X is a positive instance of the distributional tiling, then for *a.e.* G produced by the randomized reduction f, G is colorable.

Next, we show that if G can be colored, under the constrain R, to obtain a coloration (C', b) such that $C' \subseteq C$, then the coloration will enforce G to be colored that represents a tiling pattern. This implies that X is a positive instance of the distributional tiling. Moreover, the tiling pattern can be obtained easily from the coloring. To see this, we first note that the blank edges are exactly those specified in R, which lay down the framework for simulating tiling. Note that t_k, v_1, z_1 is a unique triangle due to the selection of v_1 by decreasing code. Since t_k has to be colored blank, this triangle enforces the beginning of the tiling simulation. Note that triangles $v_i \xrightarrow{\mathbf{I}} z_i \xrightarrow{\mathbf{I}} v_{i+1}$ will enforce the initial sequence S, where the presence or absence of an edge between v_i to v_{i+1} depends on S. Hence, G is forced to be colored to represent an $m \times m$ tiling pattern. This proves the following lemma.

Lemma 6.13 *For a.e. random string $s \in \Gamma'(X)$, X is a positive instance of the distributional tiling problem iff the graph G produced by $f(X, s)$ is colorable. Moreover, the tiling can be constructed in polynomial time from a coloring.*

The reduction $f(X, s)$ we described above is clearly one-one and p-time computable since the embedding is one-one and p-time computable. Also, although Γ may not be certifiable, a solution to the coloring can be transformed to a solution to the tiling by the reduction, and so the correctness of the yes/no answer to X can be efficiently verified to perform the iteration (see Theorem 6.1).

We are only left to verify that the domination condition is satisfied. Note that the size of R is proportional to $\log m$, and so $|R|^{-2}2^{-|R|}$ is proportional to $n^{O(1)}$. Hence, the probability of $f(X, s)$ is proportional to $(2n + k)^{-4}2^{-(2n+k)^2}n^{O(1)}$. By definition, $\mu_\Gamma(X, s) = \mathbf{Pr}[X]2^{-|s|} = \mathbf{Pr}[X]2^{-m^7} < 2^{-m^7}$, where $\mathbf{Pr}[X]$ is the probability distribution of X. Since $k = O(\log n)$ and $n < 8m^{3.45}$, $-m^7 < -(2n + k)^2$ when m is sufficiently large. Also note that $n > 2m^{3.4}$. Hence, $\mu_\Gamma(X, s)$ is dominated by the distribution of $f(X, s)$. This completes the proof. ∎

6.4 Distributional Matrix Correspondence

The distributional matrix correspondence problem was first shown to be complete for DistNP by Gurevich [Gur90]. A detailed proof was given in [BG95]. The proof presented here is based on [BG95].

Recall that we use $\mathrm{SL}_2(\mathbb{Z})$ to denote the set of unimodular matrices, and a unimodular matrix X is a two-by-two matrix of integer entries such that $\det(X) = 1$. A matrix is represented by a list of its entries in binary notation. We begin with a discussion on positive unimodular matrices.

Positive matrices

A unimodular matrix is called *positive* if it has no negative entries. Positive matrices form a monoid $\mathrm{SL}_2(\mathbb{N})$ while $\mathrm{SL}_2(\mathbb{Z})$ forms a multiplicative group called *modular group*. In this section, by a column we mean a column of two relatively prime nonnegative integers. We may view a positive matrix as the pair of its columns. If u is a column, we use u_1 to represent the upper and u_2 the lower components of u. Let u and v be two columns. Write $u \leq v$ if $u_1 \leq v_1$ and $u_2 \leq v_2$, and write $u < v$ if $u \leq v$ and either $u_1 < v_1$ or $u_2 < v_2$. Define $\max(X)$ to be the maximal entry of a positive matrix X. Let

$$A_0 = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix} \text{ and } B_0 = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}.$$

Lemma 6.14 *(1)* $(u,v) \times A_0 = (u+v, v)$, *and* $(u,v) \times B_0 = (u, u+v)$. *(2) If* A_0 *is a right divisor of a matrix* (u,v) *in* $\mathrm{SL}_2(\mathbb{N})$, *then* $u > v$; *if* B_0 *is a right divisor of* (u,v) *in* $\mathrm{SL}_2(\mathbb{N})$, *then* $u < v$. *(3) If the maximal entry* m *of a matrix* (u,v) *appears in two or more places, then* $m = 1$.

Proof. Part (1) is obvious. For part (2), note that if $(u,v) = (u',v')A_0$, then $(u,v) = (u'+v', v')$ from (1), so $u > v$. For part (3), if m occurs twice in the same row or the same column, then m divides $\det(u,v) = 1$ and so $m = 1$. Otherwise, we first note that the determinant being positive implies that m cannot occur on v_1 and u_2, so the only case left is $u_1 = v_2 = m$. In this case, $1 = u_1v_2 - v_1u_2 \geq m^2 - (m-1)^2 = 2m - 1$ and so $m = 1$. ∎

Lemma 6.15 $\mathrm{SL}_2(\mathbb{N})$ *is isomorphic to* Σ^*. *The two indecomposable non-unit elements are the matrices* A_0 *and* B_0. *Moreover, if a positive matrix* (u,v) *is not the unit matrix then* $u > v$ *or* $v > u$.

Proof. Note that Lemma 6.14(2) implies that A_0 and B_0 generate a free monoid. Also note that Σ^* forms a free monoid on the operation of concatenation. So it suffices to show that every non-unit positive matrix (u,v) is a product of A_0 and B_0. Define $\ell(u) = u_1 + u_2$ and $\ell(u,v) = \ell(u) + \ell(v)$. We prove the lemma by induction on $s = \ell(u,v)$. Since the entries of the main diagonal cannot be 0, $s \geq 2$. Note that A_0 and B_0 are the only non-unit matrices of weights ≤ 3. So the case $s \leq 3$ is obvious. Suppose $s > 3$. Then $m = \max(u,v) > 1$. We assume that m occurs in u, the case that m occurs in v is similar. If $u_1 = m$, then $1/m = (u_1v_2 - v_1u_2)/m > v_2 - u_2$ and so $u_2 \geq v_2$. Similarly, if $u_2 = m$ then $u_1 \geq v_1$. Thus, the column $u - v$ has nonnegative entries and so $u > v$. Since $\det(u-v, v) = 1$, $(u-v, v) \in \mathrm{SL}_2(\mathbb{N})$. By the induction hypothesis, $(u-v, v)$ is a product of matrices A_0 and B_0. By Lemma 6.14(1), $(u,v) = (u-v, v) \times A_0$. ∎

We call the greater column of a non-unit positive matrix is called the *major* column. In the case of the unit matrix, we call either column *major*. The other column is call *minor*.

Lemma 6.16 *The major column and one bit indicating whether it is the first or the second column uniquely define the minor column. Moreover, the minor column can be found in polynomial time.*

Proof. Without loss of generality, assume that (u, v) is not the unit matrix. It follows that both components of the major column are positive. Suppose that u is the major column (the case that v is the major column is similar). We will show that v is the *least* column such that $u_1v_2 - u_2v_1 = 1$. Let w be any column such that $u_1w_2 - u_2w_1 = 1$. Then $u_1(w_2 - v_2) = u_2(w_1 - v_1) = u_1u_2k$ for some k because u_1 and u_2 are relatively prime. Hence, $w_1 = v_1 + ku_1$ and $w_2 = v_2 + ku_2$. If $k < 0$, then either w_1 or w_2 is negative. Thus, $k \geq 0$ and therefore $w_1 \geq v_1$, $w_2 \geq v_2$. The minor column v can be found in polynomial time by using the extended Euclid's algorithm [Knu73]. ∎

Remark 6.4 Instead of columns, we may also state the above results using rows. This would only cause some slight changes in Lemma 6.14 (for example, the first statement would say that $A_0 \times (u, v) = (u, u + v)$, and $B_0 \times (u, v) = (u + v, v)$, where u is the upper row of the given matrix and v is the lower row), but Lemmas 6.15 and 6.16 remain true. ∎

Let n be a natural number. Denote by $|n|$ the length of the binary notation of n. Let X be a positive matrix, and $|X| = |\max(X)|$.

Lemma 6.17 *The uniform probability* $\mathbf{Pr}(X : |X| = l) = \Theta(2^{-2l})$.

Proof. It suffices to prove that the number $N(l)$ of positive matrices of size l is $\Theta(2^{2l})$. Let $\phi(m)$ denote the number of positive integers $n \leq m$ that are prime to m, and $\Phi(m) = \phi(1) + \cdots + \phi(m)$. Then $\Phi(m) = 3m^2/\pi^2 + O(m \log m)$[HW88]. Thus, $N(l) = \Theta(\sum_{|m|=l} \phi(m)) = \Theta(\Phi(2^l - 1) - \Phi(2^{l-1})) = \Theta(2^{2l})$. ∎

There are exactly two isomorphisms of $\mathrm{SL}_2(\mathbb{N})$ to Σ^*. One takes A_0 to 0 and B_0 to 1, denoted by I, and the other one takes A_0 to 1 and B_0 to 0. Let J denote the isomorphism I^{-1} from Σ^* to $\mathrm{SL}_2(\mathbb{N})$. By an induction on the length of x, it is easy to check that if x starts with 0 (respectively, 1), then the lower (respectively, upper) row of $J(x)$ is major. Note that the size of a matrix X may not be polynomially related to the length of the corresponding string $I(X)$. For example, a matrix $|A_0^n| = \left|\begin{pmatrix} 1 & 0 \\ n & 1 \end{pmatrix}\right| = |n| \sim \log n$ whereas $|I(A_0^n)| = |0^n| = n$. This implies that I is not p-time computable. However, we can show that I is ap-time computable. Let X be a positive matrix. Denote by $\mu_M(X)$ the probability distribution of X which is proportional to $|X|^{-1}2^{-2|X|}$ by Lemma 6.17. Let $\mu_S(x)$ be proportional to $|x|^{-2}2^{-|x|}$. .

Lemma 6.18 *I is ap-time computable.*

Proof. Let X be a positive matrix, then $I(X)$ can be computed by the following recursive algorithm. If X is the unit matrix then $I(X)$ is the empty string (the identity of the monoid Σ^*). Suppose that $X = (u, v)$ differs from the unit matrix. If u is the major column, let $w = u - v$ and $z = I(w, v)$, then $I(X) = z0$. If v is the major column, let $w = v - u$ and $z = I(u, w)$, then $I(X) = z1$. This algorithm runs in time proportional to $|I(X)|$. Next, we show that $|I(X)|$ is polynomial on μ_M-average.

Suppose that x is a nonempty binary string and let $m \leq n$ be the two entries of the major row of $J(x)$. First, we note that $\frac{n}{m}$ can be uniquely represented by a finite continued fraction $[a_0, \ldots, a_l]$, where a_0 is a nonnegative integer, a_i with $0 < i < l$ is a positive integer, and a_l is an integer ≥ 2. In other words, $\frac{n}{m} = p_l/q_l$, where $p{-}1 = 1$, $p_0 = a_0$, $p_i = a_i p_{i-1} + p_{i-2}$, $q_{-1} = 0, q_0 = 1$, $q_i = a_i q_{i-1} + q_{i-2}$, $1 \leq i \leq l$. This fact is true for any positive rational number and its proof can be found numerous number theory books such as [Ste69, HW88].

Let $s(n,m) = \sum_{i=0}^{l} a_i$. Then by an induction on $|x|$, we can show that $|x| = s(n,m)$. If $|x| = 1$, then $m = n = 1$ and $s(n,m) = 1 = |x|$. Suppose $|x| > 1$. Assume that $x = y0$. The case that $x = y1$ is similar. Let (i,j) be the major row of $J(y)$. Then $(n,m) = (i+j, j)$ by Lemma 6.14 and Remark 6.4. It suffices to show that if $i \leq j$, then $s(n,m) = s(j,i) + 1$, and if $i > j$, then $s(n,m) = s(i,j) + 1$. Since $J(y)$ is not the unit matrix, neither i nor j is zero. In any case, $\frac{n}{m} = \frac{i+j}{i} = 1 + \frac{j}{i}$. If $i \leq j$, write $\frac{j}{i} = [a_0, \ldots, a_k]$, then $\frac{n}{m} = [a_0 + 1, \ldots, a_k]$, so $s(n,m) = s(j,i) + 1$. If $i > j$, write $\frac{i}{j} = [a_0, \ldots, a_l]$, then $\frac{n}{m} = 1 + 1/(i/j) = [1, a_0, \ldots, a_l]$, so $s(n,m) = s(i,j) + 1$.

Yao and Knuth [YK75] showed that, for any positive integers $m \leq n$,

$$\sum_{m=1}^{n} s(n,m) = \frac{6n(\ln n)^2}{\pi^2} + O(n \log n (\log\log n)^2) = \Theta(n(\log n)^2).$$

Let $|X| = l$, and $a(X) < b(X)$ form the major row of X. Therefore, $\sum_{X, b(X)=n} s(b(X), a(X)) = \Theta(n(\log n)^2)$. Then $\sum_{X, b(X)=n} |I(X)| = \Theta(n(\log n)^2)$ and so $\sum_{X, |X|=l} |I(X)| = 2^l \Theta(l^2 2^l)$. By Lemma 6.17, $\mu_M(X : |X| = l) = \Theta(2^{2l})$, so $\sum_{X, |X|=l} |I(X)| \mu_M(X : |X| = l) = \Theta(l^2)$. It follows that $|I(X)|$ is polynomial on μ_M-average. ∎

The isomorphism J is p-time computable but the distribution μ_M does not dominate the distribution μ_S with respect to J. Thus, J fails to reduce Σ^* to $\mathrm{SL}_2(\mathbb{N})$.

Proposition 6.19 *μ_S cannot be (weakly) dominated by μ_M with respect to J.*

Proof. Suppose, to the contrary, that μ_S is (weakly) dominated by μ_M with respect to J. Since J is one-one, there is a function g such that $g(x) = \mu_S(x)/\mu_M(J(x))$ is polynomial on μ_S-average. Thus, there is an $\varepsilon > 0$ such that

$$\sum_x \frac{g^\varepsilon(x)}{|x|} \mu_S(x) = \sum_n \sum_{x, |x|=n} \frac{g^\varepsilon(x)}{n} \frac{2^{-n}}{n^2} < \infty.$$

Hence, $\sum_{x, |x|=n} g^\varepsilon(x) 2^{-n} = o(n^3)$. We will show that this is impossible.

Let $l = |J(x)|$. By the definition of g and Lemma 6.17, $g(x) = \Theta(l^2 \cdot 2^{2l} \cdot n^{-2} \cdot 2^{-n}) \leq \Theta(2^{2l} n^{-2} 2^{-n})$. Let $s(x)$ be the sum of the entries of the major row of $J(x)$. Clearly, $s(x) = \Theta(2^l)$. Hence, it suffices to show that the expectation $E_n = \sum_x (s^2(x) 2^{-n})^\varepsilon 2^{-n}$ is not bounded by any polynomial of n. We may restrict our attention to $\varepsilon < 1/2$. Let $|x| = n$ and y range over strings of length $n - 1$.

Define $A(y) = \frac{1}{2}[s^{2\varepsilon}(y0) + s^{2\varepsilon}(y1)]/s^{2\varepsilon}(y)$. Then there exists an $\alpha > 1$ such that for every y, $A(y) \geq \alpha 2^{\varepsilon}$. To see this, let $a > b$ be the two entries of the major row of $J(x)$, and $\gamma = b/a$. Then

$$A(y) = \frac{1}{2} \cdot \frac{(2a+b)^{2\varepsilon} + (a+2b)^{2\varepsilon}}{(a+b)^{2\varepsilon}} = \frac{1}{2} \cdot \frac{(2+\gamma)^{2\varepsilon} + (1+2\gamma)^{2\varepsilon}}{(1+\gamma)^{2\varepsilon}}.$$

Let $\delta = 1/(1+\gamma)$. Then $A(y) = \frac{1}{2}[(1+\delta)^{2\varepsilon} + (2-\delta)^{2\varepsilon}]$. Consider the function $f(t) = t^{2\varepsilon}$. Note that $\varepsilon < 1/2$, so $f''(t) = 2\varepsilon(2\varepsilon - 1)t^{2\varepsilon - 2} < 0$. Thus, f is concave. Since $\delta \in (0,1)$, the chord C between the points $(1+\delta, f(1+\delta))$ and $(2-\delta, f(2-\delta))$ lies strictly above the chord C_0 between the points $(1, f(1))$ and $(2, f(2))$. Clearly, the center of the interval $[1+\delta, 2-\delta]$ coincides with the center 1.5 of the interval [1,2], and so the center of C lies directly above the center of C_0. Note that the arithmetical mean $(1 + 2^{2\varepsilon})/2$ of numbers 1 and $2^{2\varepsilon}$ exceeds their geometrical mean 2^{ε}. Thus, $A(y) > (1 + 2^{2\varepsilon})/2 > 2^{\varepsilon}$. The desired α is $(1 + 2^{2\varepsilon})/2^{1+\varepsilon} > 1$. Thus,

$$\begin{aligned} E_n &= \sum_x \left(\frac{s^2(x)}{2^n}\right)^{\varepsilon} 2^{-n} = 2^{-\varepsilon n} \sum_x \frac{s^{2\varepsilon}(x)}{2^n} \geq 2^{-\varepsilon n} \cdot \sum_y \frac{s^{2\varepsilon}(y) A(y)}{2^{n-1}} \\ &\geq 2^{-\varepsilon n} \sum_y s^{2\varepsilon}(y) \cdot \alpha 2^{\varepsilon} \cdot 2^{-(n-1)} = \alpha \sum_y (s^2(y) 2^{-(n-1)})^{\varepsilon} 2^{-(n-1)} \\ &= \alpha E_{n-1}. \end{aligned}$$

It follows that $E_n \geq \alpha^n$ ($\alpha > 1$) and therefore is not bounded by any polynomial of n. ∎

Positive Matrix Correspondence

Let M be a nondeterministic Turing machine. Let $K(M) = \{(x,n) : M \text{ accepts } x \text{ in } n \text{ steps}\}$, and $\mu_K(x,n)$ be proportional to $|x|^{-2}2^{-|x|}n^{-2}$. The size of (x,n) is $|x| + n$. From the proof of Theorem 4.1, it is easy to see that there is a Turing machine T such that $(K(T), \mu_K)$ is complete for DistNP. Moreover, we can assume that T accepts x iff T halts on x. Let $\xi(x,n)$ be a distribution proportional to $\mu_M(J(x))n^{-2}$.

Lemma 6.20 *There is a nondeterministic Turing machine U such that $(K(U), \xi)$ is complete for* DistNP *under a p-time randomized reduction, and moreover, U accepts x iff U halts on x.*

Proof. We will reduce $(K(T), \mu_K)$ to an appropriate $(K(U), \xi)$. One might be tempted to simply take $U = T$ and to use the identity function as a reduction. By Proposition 6.19, this approach fails to meet the domination requirement.

For every binary string x, let $N(x)$ be the positive integer with binary representation $1x$. If $N(x) = k$, let $S(k) = x$. We construct U such that, on any input y, U first computes $x = S(\max(J(y)))$, then simulates M on x.

Define a good-input domain Γ such that for all (x,n),

$$\Gamma(x,n) = \{s : |s| \geq |x| - 1,\ N(s) \leq N(x), \text{ and } \gcd(N(s), N(x)) = 1\},$$

where $\gcd(i,j)$ represents the greatest common divisor of i and j. Clearly, Γ is certifiable. Let $\phi(k)$ be the number of positive integers that are relatively prime to and less than k. Then $\phi(k) = \Omega(k/\log\log k)$ [HW88]. Note that if $\gcd(j,k) = 1$, then so is $\gcd(j, k-j)$ provided that $k > j$. Hence, half of the integers counted by $\phi(k) \geq k/2$, and so $|\Gamma(x,n)| \geq \frac{1}{2}\phi(N(x))$. Thus,

$$\sum_{s\in\Gamma(x,n)} 2^{-|s|} \geq \sum_{s\in\Gamma(x,n)} 2^{-|x|} = \Omega(\frac{1}{\log\log N(x)}) = \Omega(\frac{1}{\log|x|}).$$

Therefore, the rarity function $U_\Gamma(x,n) \leq O(\log|x|)$, which implies that Γ is nonrare.

By Lemma 6.16, for each $(x,n,s) \in \Gamma$, there is a unique positive matrix $A(x,s)$ with $(N(x), N(s))$ being its first and major column. Define the reduction f by

$$f((x,n),s) = (y, t(x,s)+n),$$

where $y = I(A(x,s))$ and $t(x,s)$ is the time that U needs to convert y into x.

By the definition of U, it is easy to verify that T halts on x iff T halts on x in n steps iff U halts within $t(x,s)+n$ steps on y iff U halts on y. So for all $(x,n,s) \in \Gamma$, $(x,n) \in K(T)$ iff $f((x,n),s) \in K(U)$.

Next, we check that f is ap-time computable with respect to μ_Γ. Since $y = I(A(x,s))$, y can be computed in time polynomial on μ_Γ-average. Now we consider $t(x,s)$, which is the time that U needs to compute $x = S(\max(J(y)))$ from y. Clearly $t(x,s)$ is bounded by a polynomial of $|y|$ since J, max, and S are all p-time computable. It follows that $t(x,s)$ is polynomial on μ_Γ-average.

We now check that f satisfies the domination property. Note that f is one-one and so we can use Lemma 2.1. We have $\mu_M(f((x,n),s)) = \mu_M(y, t(x,s)+n) = \Theta(\mu_M(J(y))/(t(x,s)+n)^2)$. Note that $\mu_M(J(y)) = \mu_M(A(x,s)) = \Theta(|x|^{-2}2^{-2|x|})$, $\mu_\Gamma(x,n,s) = \mu_K(x,n)2^{-|s|} = \Theta(|x|^{-2}2^{-|x|}n^{-2}2^{-|s|})$, and $|s| \geq |x| - 1$, so $\mu_\Gamma \preceq \mu_M \circ f$. ∎

Similar to the proof of Theorem 5.2, it is straightforward to reduce $(K(U), \xi)$ to the distributional positive matrix correspondence problem by taking $M = U$ in that proof. The reduction is given by $f(x) = ((J(z)\&!q_0, L'(J(z)), q(|X|))$, where $x = y01^n$ represents the instance of (y,n) of $K(U)$. The isomorphism J carries out the entire proof into the matrix setting. This provides a proof to the following theorem.

Theorem 6.21 *The distributional positive matrix correspondence problem is complete for* DistNP.

Matrix Correspondence

We now turn our attention to unimodular matrices that may or may not be positive. The technical terms we used to describe positive matrices such as major columns, major rows, and the size of the matrix will apply to unimodular matrices with respect to absolute values. In this part, we use $|\cdot|$ to represent absolute value unless otherwise stated. Let u be a column. Denote by $|u|$ the column of $|u_1|$ and $|u_2|$, and $\max(u) = \max(|u_1|, |u_2|)$. Let $X = (u,v)$ be a unimodular matrix. Write $\max(X) = \max(\max(u), \max(v))$.

Lemma 6.22 *Let $X = (u, v)$ be a unimodular matrix. Then*

1. *u_1v_2 is positive iff u_2v_1 is positive. If the are both positive, then $|u_1v_2| - |u_2v_1| = 1$, and if they are both negative, then $|u_2v_1| - |u_1v_2| = 1$.*
2. *If X is not one of the following four matrices $I = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$, $\begin{pmatrix} -1 & 0 \\ 0 & -1 \end{pmatrix}$, $\begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}$, $\begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}$, then either $|u| > |v|$ or $|u| < |v|$.*
3. *If $m = \max(u, v) > 1$, then (u, v) has exactly one entry with value m.*

Proof. 1. If one of the numbers u_1v_2 and u_2v_1 is positive and the other is negative, then $|u_1v_2 - u_2v_1| \geq 1+1$, which is impossible. If the two numbers are both positive, then $|u_1v_2|-|u_2v_1| = u_1v_2-u_2v_1 = 1$; otherwise, $|u_2v_1|-|u_1v_2| = u_1v_2 - u_2v_1 = 1$.

2. Suppose that X is not one of the four matrices listed, and suppose that neither $|u| > |v|$ nor $|v| > |u|$. Without loss of generality, assume that $|u_1| > |v_1|$ and $|u_2| < |v_2|$. (The other case is similar.) Clearly, either $u_2 \neq 0$ or $v_1 \neq 0$. Hence, by assumption, $|u_1v_2| - |u_2v_1| \geq (|v_1| + 1)(|u_2| + 1) - |u_2v_1| = |u_2| + |v_1| + 1 \geq 2$, a contradiction.

3. Suppose, to the contrary, that (u, v) has two or more entries with value m. If m occurs in both entries in the same row or in the same column, then m divides $\det(u, v) = 1$, which is impossible. Thus, suppose that m occurs exactly twice in different columns and different row. Without loss of generality, assume that it occupies the second diagonal, *i.e.*, $|u_2| = |v_1| = m$. (The other case is similar.) If $u_2v_1 > 0$, then the determinant is negative, which is a contradiction. Hence, $u_2v_1 < 0$. By part 1 above, $u_1v_2 \leq 0$ and $1 = |u_2v_1| - |u_1v_2| \geq m^2 - (m-1)^2 = 2m - 1 \geq 3$, a contradiction. ∎

Lemma 6.23 *Let $X = (u, v)$ be a unimodular matrix with $\max(X) > 1$. If u (respectively, v) is the major column of X, then there is exactly one additional matrix of the form (u, v') (respectively, (u', v)), where the column v' (respectively, u') is minor. Moreover, $v' = v \pm u$ (respectively, $u' = u \pm v$). If the major column is positive or negative, then one of the two possible minor columns is positive and the other one is negative.*

Proof. We first show that for every two unimodular matrices (u, v) and (u, v'), there is an integer k such that $v' = v + ku$. Suppose $u_1 = 0$ (the case that $u_2 = 0$ is similar). Then $-u_2v_1 = 1 = -u_2v_1$, and so $|u_2| = 1$. Let $k = (v'_2 - v_2)/u_2$ to obtain the claim. Suppose $u_1u_2 \neq 0$, then $u_1v'_2 - u_2v'_1 = 1 = u_1v_2 - u_2v_1$, and do $u_1(v'_2 - v_2) = u_2(v'_1 - v_1) = u_1u_2k$ for some k. The claim is therefore obvious.

To prove the lemma, it suffices to consider the case when u is the major column. (If instead v is the major column, then the unimodular matrix $(-v, u)$ will have the major column on the left.) Moreover, it suffices to consider that u is positive, for otherwise, the unimodular matrix $(-u, -v)$ will have the major column on the left and the major column is positive. Let u_i be the major entry of u, *i.e.*, $u_i = \max(X)$. Let (u, v') be another matrix with major column u. Then as seen above, there is a k such that $v' = v + ku$. Since $u_i > 1$, $v_i \neq 0$. If $v_i > 0$, then $k = -1$; if $v_i < 0$, then $k = 1$. Note that if $v_i > 0$ (respectively,

$v_i < 0$) then u is the major column of $(u, v-u)$ (respectively, $(u, v+u)$). Since by assumption that u is positive, $u_1 > 0$ and $u_2 > 0$. By Lemma 6.22(1), v is either positive or negative. If v is positive (respectively, negative), then v' is negative (respectively, positive). ∎

The size of a unimodular matrix X, denoted by $|X|$, is defined to be the length of $\max(X)$ in binary notation. We assume that all unimodular matrices of the same size have an equal chance to be selected.

Lemma 6.24 *Let X be a random unimodular matrix with $\max(X) = m > 1$. Then the conditional probability that X is positive is $\frac{1}{8}$. Thus, the probability that a 2×2 integer matrix X is unimodular is proportional to $\Theta(|X|^{-2} 2^{-2|X|})$.*

Proof. Let S_0 be the collection of unimodular matrices X with $\max(X) = m$. Let S_1 be the collection of matrices $X \in S_0$ such that the major row of X is positive. For every $(u, v) \in S_0$, exactly one of the two matrices (u, v) and $(-u, -v)$ belongs to S_1. It remains to show that the probability of a random matrix from S_1 being positive is $\frac{1}{4}$. Since the major component of an S_1 matrix is greater than 1, the minor component of the major column cannot be zero. Let S_2 be the collection of S_1 matrices such that the minor component of the major column is positive. For every S_1 matrix X, let X' be the result of multiplying by -1 the diagonal of X that contains the minor component of the major column. Exactly one of the two matrices X, X' belongs to S_2. It follows that S_2 contains exactly one half of the elements of S_1. By Lemma 6.23, the probability of a random S_2 matrix being positive is $\frac{1}{2}$. ∎

Clearly, the identity function reduces the distributional positive matrix correspondence problem to the distributional matrix correspondence problem.

Theorem 6.25 *The distributional matrix correspondence problem is complete for* DistNP.

6.5 Distributional Matrix Transformation

The distributional matrix transformation problem was first shown to be complete for DistNP by Gurevich [Gur90] (see also [BG95]). The proof presented here is based on the proof given in [BG95]. Recall that a linear transformation is a function $T : \mathrm{SL}_2(\mathbb{Z}) \to \mathrm{SL}_2(\mathbb{Z})$ such that $T(\sum X_i) = \sum T(X_i)$ whenever all the X_i and $\sum X_i$ are unimodular matrices. We first show that a linear transformation can be represented by a 4×4 integer matrix, and that given a 4×4 integer matrix, it is decidable in polynomial time whether it represents a linear transformation.

Let $\mathcal{C}$ be one of the following class of numbers: $\mathbb{Q}$ (the rationals), $\mathbb{R}$ (the real numbers), and $\mathbb{C}$ (the complex numbers). Let $M_2(\mathcal{C})$ denote the vector space of two-by-two matrices with entries in $\mathcal{C}$. The definition of linear transformations on $M_2(\mathcal{C})$ is the standard one. Namely, a linear transformation on $M_2(\mathcal{C})$ is a function $T : M_2(\mathcal{C}) \to M_2(\mathcal{C})$ such that for all X, $Y \in M_2(\mathcal{C})$ and $\alpha \in \mathcal{C}$, $T(\alpha X) = \alpha T(X)$ and $T(X + Y) = T(X) + T(Y)$. Let $\mathrm{SL}_2(\mathcal{C})$ denote the

multiplicative group of two-by-two matrices X with entries in $\mathcal{C}$ such that $\det(X) = 1$. For any two-by-two matrix X, write $X = \begin{pmatrix} x_1 & x_2 \\ x_3 & x_4 \end{pmatrix}$.

Lemma 6.26 (Linear Transformation Lemma) *Let T be a linear transformation of $\mathrm{SL}_2(\mathbb{Z})$. Then*

1. *T can be uniquely extended to a linear transformation $\tilde{T}$ on $M_2(\mathbb{C})$ such that $\tilde{T}(X) = T(X)$ for $X \in \mathrm{SL}_2(\mathbb{Z})$. Moreover, for every $X \in M_2(\mathbb{C})$, $\det(\tilde{T}(X)) = \det(X)$.*
2. *There exist B and C in $\mathrm{SL}_2(\mathbb{Z})$ such that either $(\forall X \in M_2(\mathbb{C}))[T(X) = BXC]$ or $(\forall X \in M_2(\mathbb{C}))[T(X) = BX^tC]$, where X^t represents the transpose of X.*

Proof. 1. Without loss of generality, we assume that $T(I) = I$.[7]

Next, we show that T can be extended to a linear transformation $\tilde{T}$ on $M_2(\mathbb{Q})$. Let $\mathcal{B} = \left\{ \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}, \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}, \begin{pmatrix} 1 & 0 \\ -1 & 1 \end{pmatrix} \right\}$. The standard basis for $M_2(\mathcal{C})$ ($\mathcal{C} \in \{\mathbb{Q}, \mathbb{R}, \mathbb{C}\}$) is the set of the following four matrices:

$$I_1 = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix},\ I_2 = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix},\ I_3 = \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix},\ I_4 = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}.$$

It is an easy exercise to verify that any of these matrices is a linear combination (with integer coefficients) of the matrices in $\mathcal{B}$. Thus, $\mathcal{B}$ is also a basis for $M_2(\mathcal{C})$. The linear extension $\tilde{T}$ on $M_2(\mathbb{C})$ is the linear transformation that agrees with T on the four matrices in $\mathcal{B}$. Note that for $X \in \mathrm{SL}_2(\mathbb{Z})$, $T(-X) = -T(X)$ because $X = -X + X + X$, and so it is easy to see that $\tilde{T}$ agrees with T on $\mathrm{SL}_2(\mathbb{Z})$ and that $\tilde{T}$ is unique.

From now on, we write T not only for the given linear transformation but also for $\tilde{T}$. It is easy to see that if a matrix $X \in M_2(\mathbb{C})$ has integer entries then so does $T(X)$.

Next, we show that T preserves determinants. Namely, for all $X \in M_2(\mathbb{C})$, $\det(T(X)) = \det(X)$. This is true for $X \in \mathrm{SL}_2(\mathbb{Z})$ by the definition of T. So we consider $X \in M_2(\mathbb{C})$. Note that $\det(I + rX) = 1 + r\mathrm{trace}(X) + r^2 \det(X)$, where $r \in \mathbb{C}$ and $\mathrm{trace}(X) = x_1 + x_4$. Thus, (1) there are at least two distinct nonzero $r \in \mathbb{C}$ such that $I + rX \in \mathrm{SL}_2(\mathbb{Z})$ iff (2) $\det(X) = \mathrm{trace}(X) = 0$.

Clearly, if X satisfies condition (1), then so does $T(X)$ since $I + rX \in \mathrm{SL}_2(\mathbb{Z})$ iff $T(I + rX) \in \mathrm{SL}_2(\mathbb{Z})$ iff $I + rT(X) \in \mathrm{SL}_2(\mathbb{Z})$. Let $\mathcal{M}$ be the set of matrices in $\mathbb{C}$ satisfying condition (1), or (2) equivalently. Then $\mathcal{M}$ is closed under T. Therefore, the linear span $\tilde{\mathcal{M}}$ of $\mathcal{M}$ is also closed under T. Note that the trace of every matrix in $\tilde{\mathcal{M}}$ is zero and that matrices with zero trace form a three-dimensional subspace of $M_2(\mathbb{C})$. Since $\mathcal{M}$ contains the matrices I_1, I_3, and $\begin{pmatrix} 1 & 1 \\ -1 & -1 \end{pmatrix}$, it follows that $\tilde{\mathcal{M}}$ is exactly that three-dimensional subspace. $\mathcal{M}$ is a cone in the three-dimensional vector space $\tilde{\mathcal{M}}$.

We now show that there is a $c \in \mathbb{C}$ such that for all $X \in \mathbb{C}$ with $\mathrm{trace}(X) = 0$, $\det(T(X)) = c \cdot \det(X)$. Let $x_1 = x$, then $x_4 = -x$. Let $y = x_2$ and $z = x_3$.

[7] If $T(I) \neq I$, we consider T' given by $T'(X) = T(I)^{-1}T(X)$, then T' is also a linear transformation. The conclusion about T' implies the same conclusion of T.

Then $\det(X) = -x^2 - yz$. Since $T(X)$ is linear and $\det(X)$ is quadratic, $\det(T(X)) = a_1x^2+a_2y^2+a_3z^2+a_4xy+a_5xz+a_6yz$ for some coefficients $a_i \in \mathbb{C}$. By assumption, $\text{trace}(X) = 0$. If $\det(X)$ is also equal to 0, then $\det(T(X))$ has to be zero as discussed above. Choose $y = x^2$ and $z = -1$ make $\det(X) = 0$ and so $\det(T(X)) = 0$. It follows that $a_1x^2 + a_2x^4 + a_3 + a_4x^3 - a_5x - a_6x^2 = 0$ for all x. It follows that $a_1 = a_6$ and $a_2 = a_3 = a_4 = a_5 = 0$. Hence, $\det(T(X)) = a_1x^2 + a_1yz = -a_1 \cdot \det(X)$.

To find out the value of c, consider $X = \begin{pmatrix} 0 & 3 \\ 1 & 0 \end{pmatrix}$ with $\text{trace}(X) = 0$ and $X' = 2I + X = \begin{pmatrix} 2 & 3 \\ 1 & 2 \end{pmatrix} \in \text{SL}_2(\mathbb{Z})$. So $1 = \det(T(X')) = \det(2I + T(X)) = 4 + 2 \cdot \text{trace}(X) + c \cdot \det(X) = 4 - 3c$. It follows that $c = 1$. We have therefore shown that for all $X \in M_2(\mathbb{C})$ with $\text{trace}(X) = 0$, $\det(T(X)) = \det(X)$.

For $X \in M_2(\mathbb{C})$ with $\text{trace}(X) \neq 0$, we can write $X = cI + Y$, where $c \in \mathbb{C}$ and $Y \in \tilde{\mathcal{M}}$. As we have shown above, $\text{trace}(T(Y)) = 0$ and so $\det(T(Y)) = \det(Y)$.Thus, $\det(T(X)) = \det(T(cI+Y)) = c^2+c\cdot\text{trace}(T(Y))+\det(T(Y)) = c^2 + \det(Y) = \det(cI + Y) = \det(X)$. The completes the proof of part 1.

2. As indicated in [BG95], the proof of part 2 can also be found from [Wae35]. We consider the complex vector space $M_2(\mathbb{C})$, and we use the following basis for this space:

$$I,\ P = i(I_1 - I_4),\ Q = I_2 - I_3,\ R = i(I_2 + I_3).$$

This basis gives a very simple formula for determinant: $\det(aI+pP+qQ+rR) = a^2 + p^2 + q^2 + r^2$.

Since T is linear and preserves determinants and I, it is easy to see that $\det(X - \lambda I) = 0$ iff $\det(T(X) - \lambda I)) = 0$ and so T also preserves eigenvalues. In particular, $T(I_1)$ has eigenvalues 0 and 1. We can view $T(I_1)$ as a transformation of two component vectors, which is a projection onto some line along some other line, and so it has the form

$$L(I_1) = \begin{pmatrix} p \\ q \end{pmatrix} \begin{pmatrix} r & s \end{pmatrix} = \begin{pmatrix} pr & ps \\ qr & qs \end{pmatrix}$$

for some p, q, r, s. Since the eigenvalues are 0 and 1, the trace is 1, and so $pr + qs = 1$. Let $A = \begin{pmatrix} p & -s \\ q & r \end{pmatrix}$, then $\det(A) = 1$. Note that $A^{-1} = \begin{pmatrix} r & s \\ -q & p \end{pmatrix}$. Let $\hat{T}(X) = A^{-1}T(X)A$, then clearly $\hat{T}(I) = I$ and $\hat{T}(I_1) = I_1$. Since $P = i(2I_1 - I)$, $\hat{T}(P) = P$. Clearly, if we can show part 2 for $\hat{T}$ with matrix B, then the same result is true for T with matrix AB. So without loss of generality, we assume that T also preserves I_1 and P.

Let

$$\langle X, Y\rangle = \frac{1}{2}(\det(X + Y) - \det(X) - \det(Y)),$$

then there are $a, a', p, p', q, q', r, r' \in \mathbb{C}$ such that $\langle X, Y\rangle = \langle aI + pP + qQ + rR, a'I+p'P+q'Q+r'R\rangle = aa'+pp'+qq'+rr'$. By part 1, $\det(T(X)) = \det(X)$ for all $X \in M_2(\mathbb{C})$, so $\langle T(X), T(Y)\rangle = \langle X, Y\rangle$. Note that T also preserves I and P, so it must leave invariant the set of vectors orthogonal to both I and P,

namely, the linear span of Q and R. Hence, we have $T(Q) = qQ + rR$ for some $q, r \in \mathbb{C}$. We also have $1 = \det(Q) = \det(T(Q)) = q^2 + r^2$. There is a complex number v such that $\frac{1}{2}(v + 1/v) = q$. Note that $[\frac{1}{2}(v + 1/v)]^2 + [\frac{1}{2i}(v - 1/v)]^2 = 1 = q^2 + r^2$, so $r = \pm\frac{1}{2i}(v - 1/v)$. Replacing v with $1/v$ if necessary, we have $q = \frac{1}{2}(v + 1/v)$ and $r = \frac{1}{2i}(v - 1/v)$.

Let $u = \sqrt{v}$, and let $M = \begin{pmatrix} u & 0 \\ 0 & 1/u \end{pmatrix}$. Then $M \begin{pmatrix} x & y \\ z & w \end{pmatrix} M^{-1} = \begin{pmatrix} x & u^2 y \\ z/u^2 & w \end{pmatrix} = \begin{pmatrix} x & vy \\ z/v & w \end{pmatrix}$. So $MQM^{-1} = \begin{pmatrix} 0 & v \\ -1/v & 0 \end{pmatrix}$. Note that $T(Q) = qQ + rR = \begin{pmatrix} 0 & v \\ -1/v & 0 \end{pmatrix}$. Let $\hat{T}(X) = M^{-1}T(X)M$, then $\hat{T}(Q) = Q$. Clearly, $\hat{T}$ also preserves I and P. Similar to what we discussed before, we can simply assume that T preserves Q as well.

It follows that T fixes the subspace orthogonal to I, P, and Q. So $T(R) = rR$ for some $r \in \mathbb{C}$. Since $1 = \det(R) = \det(T(R)) = r^2$, $r = \pm 1$. If $r = 1$, then T preserves all four basis matrices, and so T is the identity. If $r = -1$, we note that $P^{-1}(-R)^t P = R$, $P^{-1} I^t P = I$, and $P^{-1} Q^t P = Q$, and so $T(X) = P^{-1} X^t P$ for all X.

We now return to the original T which is normalized to preserve I and extended to $M_2(\mathbb{C})$, and we have just shown that there is a $U \in \mathrm{SL}_2(\mathbb{C})$ such that either for all $X \in M_2(\mathbb{C})$, $T(X) = UXU^{-1}$ or for all $X \in M_2(\mathbb{C})$, or $T(X) = UX^tU^{-1}$. We also know that if the entries of X are integers then so are those of $T(X)$. We are left to show that there are B, $C \in \mathrm{SL}_2(\mathbb{Z})$ such that part 2 of the lemma holds. Without loss of generality, we assume that $T(X) = UXU^{-1}$. The other case is similar.

Write $U = \begin{pmatrix} u_1 & u_2 \\ u_3 & u_4 \end{pmatrix}$. Then since $\det(U) = 1$, $U^{-1} = \begin{pmatrix} u_4 & -u_2 \\ -u_3 & u_1 \end{pmatrix}$. Let X be the matrix with a single entry equal to 1 and all other entries zero. then the entries of $T(X)$ are all integers which are products of two entries of U. By considering all four possible locations of the entry 1, we obtain all possible products of any two entries of U, which are all integers. In particular, the square of each entry of U is an integer. So each entry of U can be written in the form abc, where a is either i or 1, $b \in \mathbb{Z}$, and c is either 1 or c is a square root of a positive, non-square number. We first show that c has to be equal to 1. If not, then c must contain $\sqrt{p}$ as a factor for some prime number p. It follows that $\sqrt{p}$ has to be a factor of every other non-zero entry because the product of one number with $\sqrt{p}$ as a factor and one without cannot be an integer. Hence, p divides $\det(U) = 1$, which is impossible. If $a = i$ in one entry, then a must be i in every non-zero entry because the product of one number with i and one without cannot be an integer. Let $B = \frac{1}{i}U$ and $C = -\frac{1}{i}U^{-1}$. If $a = 1$, then let $B = U$ and $C = U^{-1}$. This completes the proof of part 2. ∎

Let T be a linear transformation on $\mathrm{SL}_2(\mathbb{Z})$. By Lemma 6.26, T can be uniquely extended to a linear transformation on $M_2(\mathbb{C})$. For convenience, we still call this extension T. Let

$$M(T) = \begin{pmatrix} T(I_1) & T(I_2) \\ T(I_3) & T(I_4) \end{pmatrix}.$$

By Lemma 6.26(1), $M(T)$ is a four-by-four matrix with integer entries. Also,

it is easy to see that

$$L\begin{pmatrix} x_1 & x_2 \\ x_3 & x_4 \end{pmatrix} = \begin{pmatrix} y_1 & y_2 \\ y_3 & y_4 \end{pmatrix} \text{ iff } M(T)\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{pmatrix}.$$

Let X be four-by-four matrix with $\det(X) = 1$, we have $\det(M(T^{-1})) \cdot \det(M(T)) = \det(M(T^{-1})M(T)X) = \det(X) = 1$. This implies that $\det(M(T)) = \pm 1$. In Lemma 6.26(2), T is called a *right* (respectively, left) transformation in the first (respectively, second) case. Using the fact that $\det(B) = \det(C) = 1$, it is straightforward to show T is a right transformation iff $\det(M(T)) = 1$.

Lemma 6.27 *There is a p-time algorithm that decides for a given four-by-four integer matrix M whether $M = M(T)$ for some linear transformation T.*

Proof. We first calculate $\det(M)$. If $\det(M) \neq \pm 1$, then M does not represent any linear transformation. Assume that $\det(M) = 1$. The case that $\det(M) = -1$ is similar. Then M might represent a linear transformation T such that there are B and C in $\mathrm{SL}_2(\mathbb{Z})$ such that for all X, $T(X) = BXC$. Note that for any B and C in $\mathrm{SL}_2(\mathbb{Z})$, BXC is a linear transformation. We will show how to find B and C. Let $B = \begin{pmatrix} b_1 & b_2 \\ b_3 & b_4 \end{pmatrix}$ and $C = \begin{pmatrix} c_1 & c_2 \\ c_3 & c_4 \end{pmatrix}$. If M represents T, then

$$M = M(T) = \begin{pmatrix} b_1c_1 & b_1c_3 & b_2c_1 & b_2c_3 \\ b_1c_2 & b_1c_4 & b_2c_2 & b_2c_4 \\ b_3c_1 & b_3c_3 & b_4c_1 & b_4c_3 \\ b_3c_2 & b_3c_4 & b_4c_2 & b_4c_4 \end{pmatrix}.$$

We can recover all b_i/b_j and all c_i/c_j from M. Thus, we can recover B and C up to a scalar factor. Using the fact that $\det(B) = \det(C) = 1$ we can find B and C. Hence, we can determine whether $M = M(T)$ for some T in polynomial time. ∎

For convenience, we may ignore the distinction between a linear transformation T on $\mathrm{SL}_2(\mathbb{Z})$ and its matrix representation $M(T)$ due to Lemmas 6.26 and 6.27. For $B,\ C \in \mathrm{SL}_2(\mathbb{Z})$, let $T_{B,C}$ be the linear transformation such that $T_{B,C}(X) = C^{-1}XB$ for all X.

Theorem 6.28 *The distributional matrix transformation problem is complete for* DistNP.

Proof. We will reduce the distributional matrix correspondence problem to the distributional matrix transformation problem. Let $(A, L(m), n)$ be an instance of the distributional matrix correspondence. Let $L'(m)$ be the result of replacing each pair (B, C) in $L(m)$ with the linear transformation $T_{B,C}$. Define a reduction f by $f(A, L(m), n) = (A, L'(m), n)$. Clearly, $AB_1 \cdots B_k = C_1 \cdots C_k$ iff $L_{B_k,C_k} \circ \cdots \circ L_{B_1,C_1}(A) = I$.

We are only left to verify the domination property. It suffices to show that the distribution of $L_{B,C}$ dominates the distribution of (B, C). Recall

that since $\det(C) = 1$, C^{-1} has the same entries as those of C except for the signs and locations. So $\max(M(L_{B,C})) = \max(B) \times \max(C)$. Hence, $|M(L_{B,C})| = |\max(B) \times \max(C)|$. Let $l(B) = l(B)$ and $l(C) = l(C)$. Then $l(B) + l(C) - 1 \leq |M(L_{B,C})| \leq l(B) + l(C)$. Let $l = |M(L_{B,C})|$. Then the number of four-by-four matrixes M of size l is

$$\begin{aligned}
& \Theta(|\{(B,C) : l \leq l(B) + l(C) \leq l+1\}|) \\
= \; & \Theta(\sum_{j=1}^{l+1} |\{B : l(B) = j\}| \times |\{C : l - j \leq l(C) \leq l - j + 1\}|) \\
= \; & \Theta(\sum_{j=1}^{l+1} 2^{2j} 2^{2l-2j}) \text{ (by Lemmas 6.17 and 6.24)} \\
= \; & \Theta(2^{2l}(l+1)).
\end{aligned}$$

Hence, the distribution of $M(L_{B,C})$ is proportional to $l^{-2}2^{-2l}(l+1)^{-1}$. Note that the distribution of pair (B,C) is $\Theta((l(B)l(C))^{-2}2^{-2(l(B)+l(C))} \leq \Theta(l^{-2}2^{-2l})$, which is dominated by the distribution of $M(L_{B,C})$. This completes the proof. ∎

7 Final Remarks and Open Problems

Venkatesan and Rajagopalan [VR92] showed that the distributional matrix representability problem is complete for DistNP by reducing the distributional Post correspondence problem to it under a randomized reduction.[8] They also attempted to show that the bounded version of Diophantine problem is average-case NP-complete. Their approach, however, depends on a number theoretic conjecture that remains unproven. The unbounded version of the Diophantine problem is essentially Hilbert's tenth problem, which was shown to be undecidable by Matijasevič [Mat70] based on the work of Davis, Putnam, and Robinson [DPR61]. (A simplified proof of this result can be found from [JM91]). The bounded version of the Diophantine problem has been studied by Adleman and Manders [AM75, AM76, MA78].

Distributional Diophantine Problems

Instance. A positive integers d, r, and an integer polynomial $p(a, y_1, \cdots y_r)$ of degree d with r variables, where a is a free term.

Question. Do there exist non-negative integers $y_1, \ldots, y_r$ such that $|y_i| \leq |a|^{O(1)}$ for $i = 1, \ldots, r$, and $p(a, y_1, y_2, \ldots, y_r) = 0$?

Distribution. First randomly choose polynomial p and then randomly and independently choose a with respect to the default random distributions. The

[8] Gurevich [Gur90] (see also [BG95]) showed that the (worst-case) matrix representability problem on 2×2 unimodular matrices is NP-complete, and he conjectured that the distributional version of the problem is solvable in time polynomial on average. This conjecture has been proven by Cai *et. al.* [CFKL94]. The reader may find it interesting because it shows that we are working close to the boundary between AP and DistNP [Gur96]. An example in the worst-case complexity with a sharp boundary is 2SAT and 3SAT.

polynomial p may be chosen with any samplable distribution in which every polynomial has a non-zero probability.

Showing that the distributional Diophantine problem is average-case NP-complete is a challenging task, which may require a deeper understanding of number theory.

We can see that almost all the average-case NP-complete problems known so far are bounded versions of some classical undecidable problems except for the distributional graph edge coloring problem and the distributional matrix transformation problem. The task of bounding a witness size to put a problem in NP is not always trivial. A well-known example is the proof of PRIMES $\in$ NP [Pra75]. It is interesting to note that the use of randomized reductions has opened a door for us to find average-case NP-complete problems that do not belong to such kind of bounded problems.

It is an important and challenging task to systematically investigate the average-case complexity of the NP-complete problems listed in [Kar72], and in [GJ79] at large. Some of these problems have been shown easy on average, but many of them remain unknown. Reductions will certainly play an important role in this investigation. In particular, randomized reductions will be a powerful tool to show that a distributional problem is average-case NP-complete since flat distributions come quite naturally to many of these problems.

Acknowledgment. I am grateful to Jay Belanger and Yuri Gurevich for reading an early version of this paper and I thank them for their helpful comments.

References

[AM75] L. Adleman and K. Manders. The computational complexity of decision procedures for polynomials. In *Proceedings of the 16th Annual Symposium on Foundations of Computer Science*, IEEE Computer Society Press, pages 169–177, 1975.

[AM76] L. Adleman and K. Manders. Diophantine complexity. In *Proceedings of the 17th Annual Symposium on Foundations of Computer Science*, IEEE Computer Society Press, pages 81–88, 1976.

[BES80] L. Babai, P. Erdos, and M. Selkow. Random graph isomorphism. *SIAM Journal on Computing*, 9:628–635, 1980.

[BW] J. Belanger and J. Wang. No NP problems averaging over ranking of distributions are harder. *Theoretical Computer Science*, to appear.

[BW93] J. Belanger and J. Wang. Isomorphisms of NP-complete problems on random instances. In *Proceedings of the 8th Annual Conference on Structure in Complexity Theory*, IEEE Computer Society Press, pages 65–74, 1993.

[BW95] J. Belanger and J. Wang. Rankable distributions do not provide harder instances than uniform distributions. In *Proceedings of the 1st Annual International Computing and Combinatorics Conference*, vol. 959 of *Lecture Notes in Computer Science*, Springer-Verlag, pages 410–419, 1995.

[BW96] J. Belanger and J. Wang. Reductions and convergence rates of average time. In *Proceedings of the 2nd Annual International Computing and Combinatorics Conference*, vol. 1090 of *Lecture Notes in Computer Science*, Springer-Verlag, pages 300–309, 1996.

[BCGL92] S. Ben-David, B. Chor, O. Goldreich, and M. Luby. On the theory of average case complexity. *Journal of Computer and System Sciences*, 44:193–219, 1992. (First appeared in *Proceedings of the 21st Annual Symposium on Theory of Computing*, ACM Press, pages 204–216, 1989.)

[BG91] A. Blass and Y. Gurevich. On the reduction theory for average-case complexity. In *Proceedings of the 4th Workshop on Computer Science Logic*, vol. 533 of *Lecture Notes in Computer Science*, Springer-Verlag, pages 17–30, 1991.

[BG93] A. Blass and Y. Gurevich. Randomizing reductions of search problems. *SIAM Journal on Computing*, 22:949–975, 1993.

[BG95] A. Blass and Y. Gurevich. Matrix transformation is complete for the average case. *SIAM Journal on Computing*, 24:3–29, 1995.

[Bol85] B. Bollobás, *Random Graphs*, Academic Press, 1985.

[BO93] R. Book and F. Otto. *String-Rewriting Systems*. Springer-Verlag, 1993.

[Boo59] W. Boone. The word problem. *Annals of Mathematics*, 70:207–265, 1959.

[CFKL94] J.-Y. Cai, W. Fuchs, D. Kozen, and Z. Liu. Efficient average-case algorithms for the modular group. In *Proceedings of the 35th Annual Symposium on Foundations of Computer Science*, IEEE Computer Society Press, pages 143–152, 1994.

[CS96] J.-Y. Cai and A. Selman. Fine separation of average time complexity classes. In *Proceedings of the 13th Annual Symposium on Theoretical Aspects of Computer Science*, vol 1046 of *Lecture Notes in Computer Science*, Springer-Verlag, pages 331–343, 1996.

[Coo71] S. Cook. The complexity of theorem-proving procedures. In *Proceedings of the 3rd Annual Symposium on Theory of Computing*, ACM Press, pages 151–158, 1971.

[CLR90] T. Cormen, C. Leiserson, and R. Rivest. *Introduction to Algorithms*, MIT Press and McGraw-Hill, 1990.

[Dav77] M. Davis, Unsolvable problems. In Jon Barwise, editor, *Handbook of Mathematical Logic*, pages 567–594, North-Holland, 1977.

[DPR61] M. Davis, H. Putnam, and J. Robinson. The decision problem for exponential diophantine equations. *Annals of Math. Series*, 74:425–436, 1961.

[Deh11] M. Dehn. Über unendliche diskontinuierliche gruppen. *Math. Ann.*, 71:73–77, 1911.

[GJ79] M. Garey and D. Johnson, *Computers and Intractability, A Guide to the Theory of* NP-*Completeness*, W.H. Freeman and Company, 1979.

[Gur87] Y. Gurevich. Complete and incomplete randomized NP problems. In *Proceedings of the 28th Annual Symposium on Foundations of Computer Science*, IEEE Computer Society Press, pages 111–117, 1987.

[Gur89] Y. Gurevich. The challenger-solver game: variations on the theme of P =? NP. *Bulletin of the European Association for Theoretical Computer Science*, pages 112–121, 1989. Reprinted in G. Rozenberg and A. Salomaa, editors, *Current Trends in Theoretical Computer Science*, World Scientific, pages 245–253, 1993.

[Gur90] Y. Gurevich. Matrix decomposition problem is complete for the average case. In *Proceedings of the 31st Annual Symposium on Foundations of Computer Science*, IEEE Computer Society Press, pages 802–811, 1990.

[Gur91] Y. Gurevich. Average case completeness. *Journal of Computer and System Sciences*, 42:346–398, 1991.

[Gur91b] Y. Gurevich. Average case complexity. In *Proceedings of the 18th Annual Colloquium on Automata, Languages and Programming*, vol. 510 of *Lecture Notes in Computer Science*, Springer-Verlag, pages 615–628, 1991.

[Gur96] Y. Gurevich. Private communication.

[GS87] Y. Gurevich and S. Shelah. Expected computation time for Hamiltonian path problem. *SIAM Journal on Computing*, 16:486-502, 1987.

[Ike58] N. Ikeno. A six-symbol ten-state universal Turing machine. In *Proceedings of Institute of Electrical Communications*, Tokyo, 1958.

[IL90] R. Impagliazzo and L. Levin. No better ways to generate hard NP instances than picking uniformly at random. In *Proceedings of the 31th Annual Symposium on Foundations of Computer Science*, IEEE Computer Society Press, pages 812–821, 1990.

[HW88] G. Hardy and E. Wright. *Introduction to the Theory of Numbers*, Oxford University Press, 1988.

[Joh84] D. Johnson. The NP-completeness column: an ongoing guide. *Journal of Algorithms*, 5:284–299, 1984.

[JM91] J. Jones and Y. Matijasevi$\check{\text{c}}$. Proof of recursive unsolvability of Hilbert's tenth problem. *American Mathematical Monthly*, 98:689–709, 1991.

[Kar72] R. Karp. Reducibility among combinatorial problems. In R. Miller and J. Thatcher, editors, *Complexity of Computer Computation*, Plenum Press, New York, pages 85–103, 1972.

[Knu73] D. Knuth. *The Art of Computer Programming*, Vol. 1, 2nd ed., Addison-Wesley, 1973.

[Ko83] K. Ko. On the definition of some complexity classes of real numbers. *Mathematical Systems Theory*, 16:95–109, 1983.

[Lev73] L. Levin. Universal sorting problems. *Problemy Peredaci Informacii* (in Russian), 9:115–116, 1973. English translation in *Problems of Information Transmission*, 9:265–266.

[Lev86] L. Levin. Average case complete problems. *SIAM Journal on Computing*, 15:285–286, 1986. (First appeared in *Proceedings of the 16th Symposium on Theory of Computing*, ACM Press, page 465, 1984.)

[LP81] H. Lewis and C. Papadimitriou. *Elements of the Theory of Computation.* Prentice-Hall, 1981.

[Liu68] C.L. Liu. *Introduction to Combinatorial Mathematics.* McGraw-Hill, 1968.

[MS95] J. Makowsky and A. Sharell. On average case complexity of SAT for symmetric distribution. *Journal of Logic Computation*, 5:71–92, 1995.

[MA78] K. Manders and L. Adleman. NP-complete decision problems for binary quadratics. *Journal of Computer and System Sciences*, 16:168–184, 1978.

[Mar54] A. Markov. *Theory of Algorithms*, Academy of Sciences of the USSR, Moscow, 1954.

[Mar58] A. Markov. On the problem of representability of matrices. *Z. Math. Logik Grundlagen Math* (in Russian), 4:157–168, 1958.

[Mat70] Y. Matijasevič. Enumerable sets are diophantine. *Doklady Akademii Nauk SSSR* (in Russian), 191:279–282, 1970. English translation with addendum, *Soviet Mathematics*: Doklady, 11:354–357, 1970.

[Min67] M. Minsky. *Computation: Finite and Infinite Machines*, Prentice Hall, 1967.

[Nov55] P. Novikov. On the algorithmic unsolvability of the word problem in group theory. *Trudy Mat. Inst. Steklov* 44, 143(1955). English text in *Russian Translations*, 9(1958).

[Pra75] V. Pratt. Every prime has a succinct certificate. *SIAM Journal on Computing*, 4:214–220, 1975.

[Pre79] L. Priese. Towards a precise characterization of complexity of universal and non-universal Turing machines. *SIAM Journal on Computing*, 8:507–523, 1979.

[RS93] R. Reischuk and C. Schindelhauer. Precise average case complexity. In *Proceedings of the 10th Annual Symposium on Theoretical Aspects of Computer*, vol. 665 of *Lecture Notes in Computer Science*, Springer-Verlag, pages 650–661, 1993.

[Rot88] J. Rotman. *The Theory of Groups, 3rd Edition.* Wm. C. Brown Publishers., 1988.

[Sha56] C. Shannon. A universal Turing machine with two internal states. In C. Shannon and J. McCarthy, editors, *Automata Studies*, Princeton University Press, pages 157–166, 1956.

[Ste69] B. Stewart. *Theory of Numbers*, 2nd ed., The Macmillan Company, New York, 1969.

[Thu14] A. Thue. Problems über Veränderungen von Zeihenreihen nach gegebenen Regeln. *Skr. utzit av Vid Kristiania, I. Mat.-Naturv. Klasse,* 10(1914).

[Ven91] R. Venkatesan. *Average-Case Intractability.* Ph.D. Thesis (Advisor: L. Levin), Boston University, 1991.

[VL88] R. Venkatesan and L. Levin. Random instances of a graph coloring problem are hard. In *Proceedings of the 20th Annual Symposium on Theory of Computing*, ACM Press, pages 217–222, 1988.

[VR92] R. Venkatesan and S. Rajagopalan. Average case intractability of diophantine and matrix problems. In *Proceedings of the 24th Annual Symposium on Theory of Computing*, ACM Press, pages 632–642, 1992.

[Wae35] B. Vander Waerden. *Gruppen von Linearen Transformationen*, Ergebnisse Math., IV.2, Springer-Verlag, Berlin, 1935.

[Wan96] J. Wang. Average-case computational complexity theory. In L. Hemaspaandra and A. Selman , editors, *Complexity Theory Retrospective II*, Springer-Verlag, to appear in 1996.

[Wan95a] J. Wang. Average-case completeness of a word problem for groups. In *Proceedings of the 27th Annual Symposium on Theory of Computing*, ACM Press, pages 325–334, 1995.

[Wan95b] J. Wang. Random instances of bounded string rewriting are hard. *Journal of Computing and Information, Vol. 1, No. 1, Special Issue: Proceedings of the 7th Annual International Conference on Computing and Information*, pages 11–23, 1995.

[WB93a] J. Wang and J. Belanger. On average-P vs. average-NP. In K. Ambos-Spies, S. Homer, and U. Schönings, editors, *Complexity Theory—Current Research*, pages 47–67. Cambridge University Press, 1993. (First appeared in *Proceedings of the 7th Annual Conference on Structure in Complexity Theory*, IEEE Computer Society Press, pages 318–326, 1992.)

[WB93b] J. Wang and J. Belanger. Honest iteration schemes of randomizing algorithms. *Information Processing Letters*, 45:275–278, 1993.

[WB95] J. Wang and J. Belanger. On the NP-isomorphism problem with respect to random instances. *Journal of Computer and System Sciences*, 50:151–164, 1995.

[Wat61] S. Watanabe. Five-symbol eight-state and five-symbol six-state Turing machines. *Journal of the Association for Computing Machinery*, 8:476–483, 1961.

[Wil84] H. Wilf. An O(1) expected time algorithm for the graph coloring problem. *Information Processing Letters*, 18:119–122, 1984.

[YK75] A. Yao and D. Knuth. Analysis of the subtractive algorithm for greatest common divisors. In *Proceedings of National Academy of Sciences, USA*, 72:4720–4722, 1975.