Introduction to Complexity Theory Column 90

This Column  This issue’s column is “A Brief on Short Descriptions,” by Jason Teutsch and Marius Zimand. Warmest thanks to Jason and Marius! By the way, I love that the authors used a picture in which they appear together: It reminds one that results are not the only outcome of research—the research process brings people together, often forming friendships that mean quite a lot.

One of the most interesting things about theoretical computer science is how often the universe plays tricks on us. Our intuition may say things should be one way, yet the universe is being playful, and something quite different holds. Proving $P \neq NP$, for example, would not fit that bill, since intuition says that $P \neq NP$ holds… the result simply is proving hard to prove. But there is no shortage of cases where the universe has shown itself to be a surprising place. As just a few examples, let me mention: $PSPACE = IP = \text{ProbabilisticPSPACE}; NL = \text{coNL};$ the PCP Theorem; and $PH \subseteq BP \cdot \oplus P \subseteq \text{PP}$. The field can and should certainly have great pride in the fact that its members have caught and revealed these and so many other cases where the universe is being twisty.

The reason I mention the above is that the current column is, at least to me, a case where the universe is being twisty: What holds is quite surprising. But don’t take my word for it: Please read the wonderful article that Jason and Marius have crafted, and see whether you too are surprised!

Same Bat-Channel  Please stay tuned to the coming issues, whose complexity theory columns will include: Alexander Razborov (tentative title: “Proof Complexity and Beyond”), Swastik Kopparty and Shubhangi Saraf (tentative title: “Local Testing and Decoding of High Rate Error Correcting Codes”), and Neeraj Kayal and Chandan Saha (tentative topic: arithmetic circuit lower bounds).
A Brief on Short Descriptions

Jason Teutsch\textsuperscript{1} and Marius Zimand\textsuperscript{2}

Abstract

We discuss research developments on the complexity of shortest programs since the turn of the millennium. In particular, we will delve into the phenomenon of list approximation: while it's impossible to compute the shortest description for a given string, we can efficiently generate a short list of candidates which includes a (nearly) shortest description.

1 Origins and logistics

Your boss just spilled her coffee over a 1000-bit string and ruined all but the first few bits:

01101010000100111001100110011001110011001100100001000101100101111011000101

Since the well-being of your institution hinges on knowing this complete string, she asks you to recover the missing bits. You stare at the paper briefly, recognize the initial segment, and reconstruct as, “the first 1000 bits in the binary expansion of $\sqrt{2} \mod 1$.” This phrase gives an approximately shortest description of the 1000-bit string that is consistent with the one on the coffee-stained paper. Under default circumstances people prefer short descriptions, while philosophers and physicists alike speculate that nature itself likes short descriptions.\textsuperscript{3} Therefore your boss has reason to believe that you got the right string.

The story of optimal descriptions begins before computer science and even precedes the modern scientific age.

“\textit{Entia non sunt multiplicanda praeter necessitatem.}”

(Entities must not be multiplied beyond necessity.)

\textsuperscript{1}School of Computing, National University of Singapore, Computing 1, 13 Computing Drive, Singapore 117417. \texttt{teutsch@comp.nus.edu.sg}

\textsuperscript{2}Department of Computer and Information Sciences, Towson University Towson, MD 21252, USA. \texttt{mzimand@towson.edu}

\textsuperscript{3}“Nature operates in the shortest way possible.” wrote Aristotle in Book V of the Physics \cite{26}. Newton \cite{19} states as Rule I of his Rules Reasoning in Philosophy, “We are to admit no more causes of natural things than such as are both true and sufficient to explain their appearances.” He further explains, “…Nature is pleased with simplicity, and affects not the pomp of superfluous causes.” Galileo wrote, “…Nature does not multiply things unnecessarily; that she makes use of the easiest and simplest means for producing her effects…” \cite{25}. 
This popular Latin maxim, often attributed to a 14th-century friar named William of Ockham [68], reflects a principle today known as Occam’s Razor. It expresses a preference for simplicity over excessive complexity and has excelled as a guiding scientific principle for hundreds of years.

Computer science demands a precise formulation of Occam’s Razor, as highlighted by the following example. Berry’s Paradox, named after G. G. Berry of the Bodleian Library, first appeared in Whitehead and Russell’s book *Principia Mathematica* [71] early in the twentieth century:

The number of syllables in the English names of finite integers tends to increase as the integers grow larger, and must gradually increase indefinitely, since only a finite number of names can be made with a given finite number of syllables. Hence the names of some integers must consist of at least nineteen syllables, and among these there must be a least. Hence “the least integers not nameable in fewer than nineteen syllables” must denote a definite integer; in fact, it denotes 111,777. But “the least integer not nameable in fewer than nineteen syllables” is itself a name consisting of eighteen syllables; hence the least integer not nameable in fewer than nineteen syllables can be named in eighteen syllables, which is a contradiction.

The authors included this example in a list of seven paradoxes illustrating the “vicious-circle fallacy” foundational crisis in mathematics. With the emergence of computer science still on the horizon, the authors simply dodge Berry’s Paradox and the other paradoxes by means of their “theory of logical types.”

In the 1960s Solomonoff, Kolmogorov and Chaitin formalized the notion of “description” in terms of computer programs, thereby offering a more elegant resolution to Berry’s Paradox. Fix a standard machine $U$ for Kolmogorov complexity. A string $p$ is a description or program for $x$, if $U(p) = x$. The Kolmogorov complexity of a string $x$ is

$$C(x) = \min\{|p| : U(p) = x\},$$

where $|p|$ denotes the length of string $p$. Kolmogorov complexity has numerous applications in program analysis [44, 70], algorithmic randomness [17], learning theory [31, 42], phylogeny [11], and empirical animal behavior experiments [55]. We will not attempt to cover this wide spectrum of activity in the survey paper, and instead refer the interested reader to the standard book by Li and Vitányi [41]. Neither will we give a comprehensive survey of sets of minimal programs, as Schaefer did that in his Master’s Thesis [56]. Rather we will focus on three specific areas of recent development: post-Schaefer results on the complexity of sets of minimal programs (Section 2), list approximations (Section 3), and complexity of distributed compression (Section 4).

Section 3 focuses on a recently-discovered and efficient approximation scheme for short descriptions. These results challenge conventional wisdom which says that shortest computer programs are not amenable to effective approximations because Kolmogorov complexity is computationally inaccessible. Indeed:

---

4See [41] Section 1.13 for historical details.
5A machine $U$ is called standard if for every further machine $V$ there exists a polynomial-time computable “translator” function $t$ and a constant $d$ such that for all descriptions $p$, $U(t(p)) = V(p)$ with $|t(p)| \leq |p| + d$. The polynomial-time results in this paper require both effective and efficient translators which are not assumed in the usual universal machines for Kolmogorov complexity. See [54] Section 3.1.6 for a detailed, explicit construction of a standard machine.
6We will also use the the word “program” to refer to the index of a p.c. function in a given numbering (as defined in Section 2.1). We permit the word “program” to refer to both strings and functions, and whether we refer to either or both will be clear from context.
1. any algorithm can enumerate at most finitely many strings of high complexity \[41\];
2. no unbounded computable function is a lower bound for Kolmogorov complexity \[74\];
3. Any enumerator \( A \) satisfying \( C(x) \in A(x) \) must satisfy \( |A(x)| = \Omega(|x|) \) for all but finitely many \( x \) \[4\]; and
4. the set of Kolmogorov random strings is not \((1,k)\)-recursive for any \( k \) \[67\].

Is it possible to effectively determine on input \( x \) a polynomial-size list containing a short description of \( x \)? For descriptions whose length is within \( O(1) \) bits of optimal, the answer is yes! In fact, we can even do this in polynomial-time.

Section 4 is dedicated to the problems of seeking short descriptions for two or more correlated strings. For example, consider the following situation. Alice knows a line \( \ell \) in the affine plane over \( \mathbb{F}_2^n \), Bob knows a point \( P \) on \( \ell \), and they want to send \( \ell \) and \( P \) to Zack. Alice has \( 2n \) bits of information: the slope and the intercept of \( \ell \). Bob also has \( 2n \) bits of information: the two coordinates of \( P \). However, together they have only \( 3n \) bits of information. We assume that Alice, Bob, and Zack know the correlation between \( \ell \) and \( P \), namely that \( P \) is on \( \ell \). Alice can send \( 2n \) bits (the entire \( \ell \)), and then Bob only needs to send \( n \) bits (one coordinate of \( P \)). It is also possible that Bob sends \( 2n \) bits (the entire \( P \)) and Alice sends \( n \) bits (say, the slope of \( \ell \)). Clearly, together Alice and Bob need to send \( 3n \) bits, and each of them needs to send at least \( n \) bits. But is it possible that Alice and Bob each send \( 1.5n \) bits? Is it possible that Alice sends \( 1.74n \) bits and Bob sends \( 1.26n \) bits? We will see that the answer to both of these questions is essentially yes, modulo some small overhead in the length of messages.

We now turn our minds back to the twentieth century as we examine the world of computability theory through the focused, modern lens afforded by sets of minimal programs.

## 2 The power of collective: sets of minimal programs

Sets of minimal programs serve as concise examples of complex objects. Developments in computability theory help make this statement precise. Indeed, many objects once only known to exist via complex constructions now wear the familiar face of minimal programs. In the 1930’s, for example, Gödel’s Incompleteness Theorem represented a major scientific and philosophical breakthrough. While Gödel’s original proof was hard \[27\], today we can easily prove the following version of his result by exploiting the following set:

\[
\text{RAND} = \{ x : C(x) \geq |x| \}.
\]

Informally, the Incompleteness Theorem says that not every true mathematical statement is provable. Fix a language sufficiently expressive so as to include all computable predicates, logical connectives, and universal quantifiers.

**Incompleteness Theorem.** Any set of true, computable axioms cannot prove the statement “\( x \in \text{RAND} \)” for sufficiently long, random strings \( x \).

**Proof.** Since a computer program can list all valid derivations, any set of theorems derivable from a computable set of axioms is computably enumerable. On the other hand, no computably enumerable set lists more than finitely many random strings. To wit, suppose that some algorithm enumerated infinitely many random strings. Then we could use this algorithm to obtain a computable function \( f \) which maps each integer \( n \) to a random string of length at least \( n \). Now \( f(n) \) can be described
in \(\log n + O(1)\) bits, contradicting that \(f(n)\) is random for all sufficiently large \(n\). Thus the set of valid theorems derivable from the given axioms contains only finitely many statements of the form “\(y \in \text{RAND}\).”

A set is called **immune** \([13]\) if it is infinite but contains no infinite, computably enumerable (c.e.) subsets.\(^7\) The astute reader may notice that any immune set with c.e. complement suffices to obtain an incompleteness theorem like the one above, but prior to Barzdin’s observation that the set RAND has these properties \([74]\), we lacked a “simple” example of such a set. The idea of using Kolmogorov complexity to obtain an incompleteness theorem is due to Chaitin \([9]\).

### 2.1 Numberings

Many properties of minimal programs depend on the underlying programming language, or *numbering*, used to describe them. Remarkably, however, some do not! In order to achieve robust results, we require numberings to satisfy certain basic properties. First, numberings should be **universal** in the sense that that they include all partial-computable (p.c.) functions. Given a numbering \(\varphi\), which is formally itself a p.c. function from \(\mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}\) and denoted as \(\varphi_e(x) = \varphi(e, x)\), the sequence \(\varphi_0, \varphi_1, \varphi_2, \ldots\) constitutes the programs of \(\varphi\). If programs from any given numbering can be effectively translated into programs for \(\varphi\), then we say \(\varphi\) is **acceptable**, i.e., for any further numbering \(\psi\), there exists a computable function \(f\) such that \(\varphi_{f(e)} = \psi_e\) for all \(e\). If the function \(f\) which witnesses that \(\varphi\) is acceptable happens to be bounded by some linear function, then \(\varphi\) is called a **Kolmogorov** numbering. This name “Kolmogorov” presumably derives from the fact that Kolmogorov numberings admit effective translation functions with constant number of bits blow up in program size, similar to the blow-up for universal machines for strings in the context of Kolmogorov complexity \([41]\).

We now pick up on the story of the set of minimal indices

\[
\text{MIN}_\varphi = \{e : (\forall j < e) [\varphi_j \neq \varphi_e]\},
\]

where Schaefer’s comprehensive survey \([56]\) left off in 1998. But first, as motivation for the above numberings properties, consider what life might be like without them. In 1958 Friedberg \([22]\) proved that there exists a numbering, now known as a **Friedberg** numbering, containing exactly one program for each p.c. function. Every Friedberg numbering \(\psi\) satisfies \(\text{MIN}_\psi = \mathbb{N}\), a decidedly uninteresting set from the computability point-of-view. The general situation, however, is not as bleak as it once seemed twenty years ago. For example, in every universal numbering, *including any Friedberg numbering*, the set

\[
\text{DMIN}_\varphi = \{e : (\forall j < e) [W^e_\varphi \neq W^j_\varphi]\}
\]

has a Turing degree satisfying \(\text{DMIN}_\varphi \oplus K \equiv_T K'\) \([32]\). Here \(W^e_\varphi\) denotes the domain of \(\varphi_e\); we shall revisit this theorem later. While robust properties of some index sets\(^8\) also persist in every universal numbering \([32, 39]\), we will in this survey focus on properties of minimal programs.

\(^7\)In the cited reference \([59]\), computably enumerable sets are called “recursively enumerable sets.”

\(^8\)For example, the problem of deciding whether two c.e. sets are disjoint, namely \(\{(i, j) : W^i_\varphi \cap W^j_\varphi = \emptyset\}\), has the same degree as the halting set in every universal numbering \(\varphi\) \([32]\). In contrast, we do not know whether the problem of determining whether one c.e. set is subset of another, \(\{(i, j) : W^i_\varphi \subseteq W^j_\varphi\}\) has fixed Turing degree under all universal numberings \(\varphi\) \([32, 39]\).
Kummer later revisited Friedberg’s existential argument and reduced it to a single-page proof \[36\]. He keenly observed that \(\text{MIN}_\varphi\), which happens to be a \(\Sigma^0_2\) set, already contains exactly one index for each p.c. function. His Friedberg numbering construction, then, exploits a computable limit-approximation common to all sets bounded by this arithmetic complexity. While the notion of “smallest Gödel number” for a given program appears in Friedberg’s original construction \[22\], the powerful idea of considering the set of all such objects still loomed on the horizon in the 1950’s.

2.2 Strong reducibilities

Post’s program, initiated during World War II \[52\], began as an attempt to pinpoint the complexity of witnesses for Gödel-like incompleteness theorems (such as the one above). Post’s seminal work \[52\] presented an incompleteness theorem based on the halting set \(K\), a set which he dubbed \textit{creative}\(^9\) and by attracting attention to this class, Post may have postponed the discovery of the arguably simpler Incompleteness Theorem above. Sets which effectively admit witnesses for incompleteness became known as \textit{productive sets} \[15\] and include the complements of creative sets but do not include RAND. Indeed, a set is productive iff it computes \(\overline{K}\) via a many-one reduction \[48\] whereas, as we shall discuss below, RAND does not even compute \(\overline{K}\) via bounded truth-table reductions \[38\].

Is every noncomputable, c.e. set as complex as the halting set? And what do we mean by “as complex?” A set of integers \(A\) \textit{Turing reduces} to a set of integers \(B\), denoted \(A \leq_T B\), if there exists an oracle machine loaded with the set \(B\) on its tape which computes \(A\). When Post first introduced the notion of Turing reduction \[52\], only two Turing \textit{degrees}, or \(\equiv_T\)-equivalence classes, for c.e. sets were known, namely the degree consisting of the recursive sets and the degree consisting of the halting set. The question of determining whether other c.e. Turing degrees exist became known as Post’s problem. As an initial stepping stone towards solving this problem, Post \[52\] introduced two stronger reducibilities: truth-table degrees (\(\leq_{tt}\)) and bounded truth-table degrees (\(\leq_{btt}\)). Friedberg and Rogers \[23\] later contributed the concept of weak truth-table degrees (\(\leq_{wtt}\)). The books \[17, Section 2.20\] and \[50, Chapter III\] contain further discussion on strong reducibilities in the context of Post’s program.

**Definition.** Let \(A\) and \(B\) be sets of integers.

1. \(A \leq_{tt} B\) if one can decide membership in \(A\) by non-adaptively constructing and then evaluating a truth table consisting of predicates of the form “\(y \in B\),” that is, if there exist computable functions \(f\) and \(g\) such that for all \(x\),

\[
x \in A \iff f(x, B(0), B(1), \ldots, B(g(x))) = 1.
\]

2. \(A \leq_{btt} B\) is as in (1) above but the truth-table \(f\) queries \(B\) in only constantly many places.

3. \(A \leq_{wtt} B\) is as in (1) above, except \(f\) may diverge when supplied with wrong input values for \(B\).

A set is T-, wtt-, tt-, or btt-complete if it has the same degree as the halting set.

---

\(^9\)In Post’s own, italicized words, “\textit{mathematical thinking is, and must remain, essentially creative}” \[52\].
Post kick-started his program by exhibiting an abstract, c.e. set which is tt-complete but not btt-complete [52]. Friedberg and Rogers observed that a T-complete c.e. set need not be wtt-complete and conjectured the existence of a wtt-complete c.e. set which is not tt-complete [23]. Lachlan finally confirmed their conjecture in 1975 [40]. Friedberg and Rogers proof appealed to Post’s hypersimple set [52], which is based on a non-trivial priority construction, and Lachlan’s separation techniques, as Odifreddi writes, are “more complicated than the ones given so far” [50, p. 344].

Behold the innate power of minimal programs. In another tour-de-force, Kummer showed that the co-c.e. set RAND is tt-complete but not btt-complete [38]. The easier part of his proof (that RAND $\not\geq_{btt} K$) follows from Post’s result that simple sets cannot be btt-complete. Next, fix a universal machine prefix machine $U$ [41], and define the real number

$$\Omega = \sum_{x \in \text{dom } U} 2^{-|x|}.$$ 

The number $\Omega$, which is sometimes called a halting probability [6, 74] is left-c.e., or recursively approximable from below, and each of its length $n$ prefixes has prefix-complexity at least $n - O(1)$. As observed in [63], the following variant of RAND is wtt-complete, but not tt-complete:

$$\{x : x \text{ is a prefix of } \Omega\}, \quad \text{(2.1)}$$

because $\Omega$ itself is wtt-complete but not tt-complete [7]. Although this result does not match Lachlan’s result, it comes close: the set in (2.1) is 2-c.e. (rather than c.e.). In general, a set is $n$-c.e. if it can be computed in the limit with at most $n$ mind changes on the enumeration for each input. Note that the completeness properties up to this point in the paragraph depend on the underlying universal machine for Kolmogorov complexity. Schaefer gave an example of an acceptable numbering $\varphi$ in which an variant of the 2-c.e. set $SD_\varphi$ below is T-complete but not wtt-complete [56, Section 3.2], but at this time we do not have a candidate for a set of minimal programs which works for all acceptable numberings.

In 1972, Meyer [43] showed that for any acceptable numbering $\varphi$, $MIN \equiv_T K'$, where $K'$ denotes the halting set relative to the halting set oracle [59], and then asked the following:

**Meyer’s Problem** ([43]). Is $MIN_\varphi \equiv_{tt} K'$ for all acceptable numberings $\varphi$?

Meyer’s Problem is a long-standing open problem, but recently a “one-dimensional” version of it was resolved. Schaefer [56] invented the following set of shortest descriptions in his Masters Thesis [56]:

$$SD_\varphi = \{e : (\forall j < e) [\varphi_j(0) \neq \varphi_e(0)]\}.$$ 

Stephan and Teutsch showed that the truth-table degree of $SD_\varphi$ depends on the underlying acceptable numbering $\varphi$ [63], although the wtt-degree of this set is fixed [56]. In fact, there exists an acceptable numbering $\varphi$ such that $SD_\varphi \equiv_{tt} \Omega$.

Friedman [20] introduced a variant of RAND which identifies complexity with maximum elements of c.e. sets rather than programs called the set of domain-random strings [61]:

$$NRW_\varphi = \{x : (\exists j < x) [\max(W_j^\varphi \cup \{0\}) = x]\}$$

He asked for the Turing degree of this set, and Stephan [61] proved that $NRW_\varphi$ is T-complete whenever $\varphi$ is a Kolmogorov numbering. Friedberg’s question remains open for the case of acceptable
numberings. Davie once proved [14] that NRW is Turing-complete for all numberings which admit computable, polynomially-bounded translation functions, but unfortunately he stored his proof on a “stiffie” disk which subsequently disappeared. We hope that someday his stiffie turns up because presently we do not know how to extend Stephan’s result beyond numberings with linearly-bounded translators. Like SD$\varphi$, the truth-table degree of NRW$\varphi$ depends on the underlying acceptable numbering $\varphi$. However, the enumeration complexity\textsuperscript{10} of these sets differ. Stephan showed that there exists an acceptable numbering $\psi$ such that NRW$\psi$ is 2-c.e. [61], whereas SD$\varphi$ is 2-c.e. in every acceptable numbering. These results are optimal in the sense that in no acceptable numbering are the complements of either of these sets 2-c.e. [61, 65]. Kolmogorov numberings present even more contrast. Indeed when $\varphi$ is a Kolmogorov numbering, then NRW$\varphi$ is not n-c.e. for any $n$ [61].

Sets of minimal programs help us make sense of structures inside truth-table degrees. Observe, for example, the following bizarre situation [63]: for every acceptable numbering $\varphi$ we have $K \geq_{tt} NRW_{\varphi}$ while at the same time $K$ and NRW$\varphi$ are btt-incomparable. The degree relation between these sets may provide even more intrigue. If it were the case that NRW$\varphi$ is truth-table complete for every Kolmogorov numbering $\varphi$, then in every Kolmogorov numbering $\psi$, NRW$\psi$ would be a “natural” example of a set in the truth-table degree of the halting set which is btt-incomparable to it. We also wonder whether the truth-table degree of SD$\varphi$ is fixed under Kolmogorov numberings $\psi$ [63]. Stephan [60] showed that every truth-table degree contains infinitely many btt-degrees, and for some degrees we can achieve this structure with sets of minimal programs. Indeed in every truth-table degree between $\Omega$ and $K$, we can take the class of witnesses for Stephan’s functions to be sets of the form NRW$\varphi$ for acceptable numberings $\varphi$ [63]. One can also build infinite truth-table antichains out of SD$\varphi$ using a carefully chosen set of acceptable numberings $\varphi$ [63].

Our discussion here would not be “complete” without mentioning Friedberg [21] and Muchnik’s [46] negative answers to Post’s problem. The two authors independently invented the finite-injury priority method in order to construct new c.e. Turing degrees [59]. Today one can achieve such sets via a single-line injury-free, or even a priority-free construction [16] [17, Section 11.1.1] by constructing a set of nested prefixes with low prefix-free complexity. While it’s possible to get a priority-free solution without appealing to Kolmogorov complexity [35], the known construction “is more complicated than the standard ‘finite injury’ proof,” [36] despite being methodologically more simple. The first author of this paper once asked Frank Stephan if he could find in the NUS library Kinber’s old result stating that there exist acceptable numberings $\varphi$ and $\psi$ such that MIN$\varphi$ and MIN$\psi$ are btt-incomparable [34]. Rather than digging for Kinber’s 13-page injury/marker construction, he promptly responded with a half page Kolmogorov complexity-based proof of this result; Stephan kindly permitted us to share his work in the external addendum of this work. This point brings us to our bold and informal conclusion:

**Injury Elimination Thesis.** Every injury construction in computability theory can be replaced by a short, priority-free example based on either minimal programs or Kolmogorov complexity.

As witnessed here, the primary methodology for achieving such a transformation is to email the injury construction to Frank Stephan and then wait for the Kolmogorov complexity proof to come back.

\textsuperscript{10}One can measure enumeration complexity in terms of the Ershov hierarchy [17, Section 2.7].
2.3 Immunity

Minimal programs can both contain and avoid containing infinite arithmetic sets of any complexity. Computably enumerable sets are sometimes called \(\Sigma^0_n\) sets because for each such set, a formula consisting of a single existential quantifier followed by a computable predicate defines its membership relation. In general a \(\Sigma^0_n\) set can be expressed with \(n\) alternating quantifiers, starting with an existential one, followed by a computable predicate. The complement of a \(\Sigma^0_n\) set is a \(\Pi^0_n\) set, and together these classes form the arithmetical hierarchy \([59]\).

John Case once asked \([8]\) his students to prove that, for every acceptable numbering \(\varphi\), the set

\[
\text{MIN}_\varphi^* = \{e : (\forall j < e) [\varphi_j \neq^* \varphi_e]\}
\]

does not contain any infinite \(\Sigma^0_2\) sets, that is, \(\text{MIN}_\varphi^\ast\) is \(\Sigma^0_2\)-immune. Here \(=^*\) denotes equality everywhere except on a finite set. It turns out that in any acceptable numbering, we can find, for every \(n\), a set of minimal programs which is \(\Sigma^0_n\)-immune but not \(\Sigma^0_{n+1}\)-immune \([62]\). Indeed, when one switches from minimal p.c. functions to minimal c.e. sets (e.g. DMIN\(_\varphi\) rather than MIN\(_\varphi\)), then any equivalence relation, including \(=^*, \equiv_T\), and so on iterating with jump operations yields a new set of minimal programs. One can even construct a set of minimal programs which neither contains, nor is disjoint from, every arithmetic set. \(\Pi_n\)-immunity for each of these same sets, on the other hand, depends on the underlying acceptable numbering \([12]\).

Post’s program originally aimed to construct sets that were “thinner” than immune sets in a different sense. An infinite set \(A\) is called hyperimmune if for any effective sequence of disjoint, finite sets \(F_0, F_1, \ldots\), there exists an \(n\) such that \(F_n \subseteq \overline{A}\). In general, while minimal programs are immune in acceptable numberings, they are not hyperimmune \([56]\). For example, an elementary combinatorial argument shows that a member of RAND exists at each length \([41]\). Minimal programs do, however, typically satisfy an intermediate immunity property, called \(\omega\)-immune \([19]\), which suffices to ensure that an arbitrary set avoids the btt-cone above the halting set.

2.4 Non-acceptable structures

Even in non-acceptable numberings, certain combinatorial structures remain in place. For example, Kummer’s Cardinality Theorem helps us gain a foothold in a notorious open problem.

**Kummer Cardinality Theorem** \([37][14]\). Let \(A\) be a set of non-negative integers, and let \(k\) be a positive integer. Suppose that there exists an algorithm which, on any input \(x_1, \ldots, x_k\), enumerates at most \(k\) integers among \(\{0, 1, \ldots, k\}\) such that one of these integers equals \(|A \cap \{x_1, \ldots, x_k\}|\). Then \(A\) is computable.

A numbering \(\psi\) is said to have the Kolmogorov property if, as in a Kolmogorov numbering, the translation function is linearly bounded, but we no longer require the translation function to be computable. In the addendum to this work, we illustrate how the above theorem resolves the following innocuous-looking question for the case of numberings with the Kolmogorov property.

---

11 Incidentally, the first problem on that same homework assignment asked for a proof using Kleene’s Recursion Theorem \([59]\) that \(\text{MIN}_\varphi\) is immune, whereas we now have an elementary proof of this fact which goes by Kolmogorov complexity \([67]\).

12 Despite a remark in \([62]\) to the contrary, it appears to still be an open problem to determine whether or not \(\Pi_n\)-immunity can vary for Kolmogorov numberings.

13 See \([51]\) for a special case of this result and \([30]\) for a survey of this area.
Schaefer’s MIN* Problem (56). Does MIN* compute the halting set for every acceptable numbering \( \varphi \)?

Let \( A \oplus B \) denote the set which codes \( A \) onto the even numbers and \( B \) onto the odd ones. Schaefer showed that \( \text{MIN}_\varphi^* \oplus K \equiv_T K'' \) for all acceptable numberings \( \varphi \), and so a positive answer to the above question would yield \( \text{MIN}_\varphi^* \equiv_T K'' \) for every acceptable numbering \( \varphi \).

Every numbering \( \psi \) with the Kolmogorov property satisfies \( \text{MIN}_\psi^* \equiv_T K'' \) and \( \text{MIN}_\psi \equiv_T \text{DMIN}_\psi \equiv_T K' \) [32], and the Kummer Cardinality Theorem-based proof [61] of NRW_\psi \equiv_T K also goes through for all \( \psi \) with the Kolmogorov property. Even without the Kolmogorov property, we find that a reasonable amount of Turing degree structure persists in any universal numbering \( \varphi \). For example, \( \text{DMIN}_\varphi \oplus K \equiv_T K' \) and \( \text{MIN}^*_\varphi \oplus K \equiv K'' \) [52]. In addition to this property of DMIN_\varphi which fails for MIN_\varphi in Friedberg numberings \( \varphi \), we can observe a further difference between minimal indices for sets and functions: there exists a universal numbering \( \nu \) which makes DMIN_\nu 1-generic and hence hyperimmune (as well as DMIN_\nu \nless_T K), whereas MIN_\varphi is never hyperimmune in any universal numbering \( \varphi \) [32].

We conclude with a brief foray into frequency computation. A set of integers \( A \) is said to be \((1,k)\)-recursive if there exists an algorithm which, for every \( k \) distinct integers, correctly identifies at least one of them as belonging or not belonging to \( A \). Schaefer asked whether, for some acceptable numbering \( \varphi \) and some \( k \), MIN_\varphi can be \((1,k)\)-recursive [56]. While the jury is still out on this one, we do know, via combinatorial-based methods, that the Kolmogorov property for \( \psi \) suffices to prevent MIN_\psi from being \((1,k)\)-recursive [67]. We also know that MIN_\varphi cannot be \((1,2)\)-recursive for any acceptable numbering \( \varphi \) [56], unlike SD_\varphi which can have this property [63]. The next section explores a more game-theoretic interaction between short programs and combinatorics.

3 List approximations and magical help

We begin our discussion on list approximations with an elementary observation. By enumerating the first string at each length which computes \( x \), we will eventually obtain a list of size at most \(|x| + O(1)\) containing a shortest description for \( x \). Indeed the machine which computes the identity function witnesses a description for \( x \) on the standard machine of size at most \(|x| + O(1)\), so our enumerative search need not inspect beyond this bound. In fact, a short, elementary argument shows that we can even enumerate the lexicographically least description for \( x \) within \(|x| + O(|x|)\) guesses [33]. Item 3 in the introduction states that the linear bound on these list sizes is optimal.

We can also view these results from another perspective. The conditional Kolmogorov complexity of a string \( x \) given string \( y \) is defined as \( C(x \mid y) = \min \{|p| : U(p,y) = x\} \) [55] and say that \( x \) is computable from \( y \) with \( k \) help bits if \( C(x \mid y) \leq k \). In this way, Kolmogorov complexity precisely gauges the amount of non-uniform information required to compute a given object. An enumerator is an algorithm \( A \) which takes \( x \) as input and, over time, enumerates a sequence of strings \( A(x) \). The following observation, which relates the quantity \( C(f(x) \mid x) \) to the approximation of \( f(x) \) via enumerators was implicit in a footnote of [1].

**Proposition.** For any functions \( f \) and \( k \), \( C(f(x) \mid x) < k(|x|) \) for all \( x \) if and only there exists an enumerator \( A \) such that for all \( x \), \( f(x) \in A(x) \) and \( |A(x)| \leq 2^{k(|x|) - 1} \).

\(^{14}\)The hidden constant depends on the underlying standard machine \( U \).

\(^{15}\)Here we implicitly code the pair \( \langle p, y \rangle \) as a string.
Proof. For the forward implication, we run $U(p, x)$ for all programs $p$ of length less than $k(|x|)$, and for the reverse implication, the help bits give the rank of $f(x)$ in the ordered sequence $A(x)$.

Let’s first use this framework to see how much help is needed to compute $C(x)$. Clearly, for every $x$ of length $n$, we have $C(C(x) \mid x) \leq C(C(x)) + O(1) \leq \log |x| + O(1)$, where the right inequality follows from the fact that $C(x) \leq |x| + O(1)$. A matching lower bound follows from the Proposition above together with Item 3 from the introduction, namely:

**Improved Gács’s Theorem ([1]).** For every $n$, there is a string $x$ of length $n$ such that $C(C(x) \mid x) \geq \log n - O(1)$.

Both Gács’s original theorem [24], which misses this tight lower bound by a factor of $\log \log |x|$, and the enumerator result in Item 3 have difficult proofs. Bauwens and Shen [1] recently gave a short game-theoretic proof of the tight result which we recount in the addendum of this paper. Thus, as expected, $C(x)$ is a hard nut to crack. Essentially, any help that is shorter than $C(x)$ is useless for computing Kolmogorov complexity.

### 3.1 Short lists for short programs

Now suppose that we are not satisfied with just getting a shortest description as time goes to infinity, but instead we want a process which terminates in finite time. Can we still build a polynomial-length list? This is clearly a much stronger requirement than enumeration. It demands that one effectively constructs a short list of candidates that is guaranteed to contain a short program for $x$. Surprisingly, if we allow programs that are within an additive constant of minimal description length and if we are content with lists of polynomial size, the answer is yes. Recall that in Section 1 we fixed our descriptions relative to a standard machine $U$. If $p$ is a program for a string $x$ and $|p| \leq C(x) + b$, we say that $p$ is a $b$-short program for $x$. In the Short List Theorems below and in Theorem 4, the constants attached to the description sizes depend on our choice $U$, but the list sizes do not.

**Theorem 1 (Optimal-Length Short List Theorem [2]).** There exists a constant $b$ and a computable function which maps each binary string $x$ to a list of size $O(|x|^2)$ containing a $b$-short program for $x$.

The quadratic estimation on the size of the list cannot be beaten: For every $b$, any computable list containing a $b$-short program must have list size $\Omega(|x|^2/(b + 1)^2)$ [2]. For some standard machines we can take the constant $b$ above to be 0, thus achieving a short list containing absolutely shortest descriptions, while in other standard machines a sufficiently small constant $b$ forces the list size to become exponential [2].

One might wonder whether one can extend Theorem 4 to shortest programs for partial-computable functions. For any Kolmogorov numbering (as defined in Section 2.1), the answer is no [69], even in the case of enumerators [33]. Finding even (the function equivalent of) an $O(n)$-short program for a function requires an enumerator with an exponential-size list [33]. For the case of acceptable numberings, the Kummer Cardinality Theorem from the last section helps us

---

16 $b + 1$ is used to handle the case $b = 0$. The constant hidden by $\Omega$ depends on the underlying standard machine for Kolmogorov complexity (which we fixed in the first section of this paper).

17 The hidden constant depends on the underlying Kolmogorov numbering.
achieve a lower bound on the enumerator list size; in an acceptable numbering, one cannot enumerate an (up to) constant-size list containing an exactly minimal index for a given function \[33\]. We do not yet know whether there exists a numbering which is friendly towards enumerations for \(*\)-minimal indices \[33\].

We now move to polynomial-time results.

**Theorem 2** (Polynomial-Time Short List Theorem \[66, 72\] 18). There exists a polynomial-time computable function which maps each binary string \(x\) to a polynomial-size list containing a length \(C(x) + O(1)\)-description for \(x\).

One method for obtaining such a list involves carefully composing the disperser graph from \[64\] with itself \[66\]. By combining the unbalanced expander from \[28\] with the disperser construction from \[64\], one can even achieve a list size of \(O(|x|^{6+\epsilon})\) for any \(\epsilon > 0\) \[66, 72\]. We outline the pseudo-random graph construction in the addendum of this paper. It is an open problem whether one can reduce the list size further; a positive answer to the following would subsume both Short List Theorems above.

**Question 3.** Does the Polynomial-Time Short List Theorem hold for quadratic-size lists?

Consider now computation of short programs if \(C(x)\) is given. The simple enumeration procedure presented at the beginning of Section 3 contains a procedure that computes a shortest program for \(x\) from the input pair \((x, C(x))\), however this algorithm runs slower than any computable function. In fact, given any computable function \(t\), if an algorithm on input \((x, C(x))\) computes a \(b(n)\)-short program for \(x\) in time \(t(n)\), then \(b(n) = \Omega(n)\), i.e., the “short” program is not that short \[3\]. Surprisingly, the situation changes when we allow probabilistic algorithms.

**Theorem 4** (\[3\]). There exists a polynomial-time probabilistic algorithm that on input \((x, C(x))\) outputs a \(O(\log^2(|x|/\epsilon))\)-short program for \(x\) with probability \((1 - \epsilon)\).

Thus computing short programs from \((x, C(x))\) is a task that can be solved probabilistically in polynomial-time, but which deterministically cannot be solved within any computable time bound! The above theorem also implies that in probabilistic polynomial time it is possible to compute a list of size \(|x| + O(1)\) that with high probability contains \(O(\log^2 |x|)\)-short programs, beating the quadratic lower bound that we have seen above for deterministic algorithms.

Note that given \(x\) and a bound \(\ell \geq C(x)\), by exhaustive search we can find a program for \(x\) of length at most \(\ell\). An important question is whether there is a polynomial-time version of this fact.

**Question 5.** Is there a probabilistic polynomial-time algorithm that on input \((x, \ell)\), provided \(\ell \geq C(x)\), returns with high probability a program for \(x\) of length \(\ell + o(|x|)\)?

We witnessed combinatorial interactions with list-size lower bounds for enumerators earlier in this section, and with sets of minimal indices in Section 2.4. We now explore the combinatorics behind the Short List Theorems. In the well-known Hall’s Marriage Problem, there are women and men, and each of the women likes some of the men. One can interpret this situation as a bipartite graph where the left-hand vertices are women, the right-hand vertices are men, and edges indicate which women like which men. Given that each woman wants a unique and unshared husband, is it possible to satisfy all the women? Hall’s Theorem gives a necessary and sufficient condition under

\[18\] achieved polynomial-time, \(O(\log n)\)-short programs.
which this is possible [29]. In the context of short lists, we will be interested in finding marriages online. An online matching up to size $2^k$, as introduced in [47] and further discussed in the next subsection, is where we know at most $2^k$ out of a large set of women will eventually enter the room, and they select husbands “on the fly” as they come in. We explore the connection with short lists in the next section. The following theorem is essentially equivalent to the Polynomial-Time Shortlist Theorem in the sense of Lemma 6 below.

Explicit Online Matching Theorem. [60] Let $\delta > 0$. For every $k \geq 0$, there exists a polynomial-time computable, bipartite graph whose left-hand vertices consist of all binary strings of length at least $k$, whose right-hand vertices number at most $2^k + \delta$, whose left-hand vertices have polynomial degree, and which admits effective online matching up to size $2^k$.

3.2 Short List Theorems: proof techniques

The proofs of the Short List Theorems are conceptually similar and basically have two parts: one which characterizes short lists containing short programs in terms of a certain type of bipartite graph defined using a 2-party game [19] and a second which constructs the necessary bipartite graph. We present a simple construction which does not achieve optimal parameters but illustrates the core ideas. In this section we convene that $n$ denotes the length of $x$ whenever the two variables appear in the same context.

Figure 1: Online vs. standard matching [17].

As we mentioned at the end of the last section, online matchings in bipartite graphs play a key role in the proof. The formal definition of online matching will be given below via a game, but basically each time a left node makes a request the matching process assigns to it an unused right node. Note that a right node can match to a single left node, but a left node may match to more than one right node. Online matching differs from offline matching. In online matching, requests from left-hand elements appear adaptively, and each matching request must be satisfied without knowledge of future requests. For example in Figure 1 any two left nodes can be matched if both are known ahead of time while online matching for two nodes can fail. Indeed, if $x$ makes the first matching request, then an adversarial second request cannot be satisfied.

In our applications, all the nodes are labeled with binary strings, and, for all $n$, all the left nodes of length $n$ have the same finite degree $d(n)$. We say that the bipartite graph has degree $d(n)$. Such a graph is computable if there exists an algorithm that on input $(x,i)$, where $i \leq d(n)$, outputs a neighbor of $x$, which is sometimes called the $i$-th neighbor of $x$. We present our combinatorial characterization of the Short List Theorems in terms of a game between a Matcher and a Requester played on a computable, bipartite graph. The Requester’s moves are requests of the form $(x,k)$,

---

19 [2] also uses this characterization to show the quadratic lower bound on list size.

20 We allow multiple edges between two given nodes.
where $x$ is a left node and $k$ is a length. He can make at most $2^k$ requests of the form $(x, k)$ and in all such requests $k \leq |x| + O(1)$ (where the constant depends on the underlying standard machine). The Matcher moves are as follows: when a request $(x, k)$ is made, he has to satisfy it with overhead $a(n)$. This means that he assigns to $x$ one of its previously unassigned right neighbors $y$ of length at most $k + a(n)$. We will consider only the case when both players use strategies computable in the bipartite graph and moves played so far. A winning strategy for Matcher is one that satisfies all the requests when playing against any computable Requester. The following lemma establishes the promised combinatorial characterization.

**Lemma 6** (based on [2]). Let $G$ be a computable, bipartite graph whose left and right sets are both the set of all binary strings. For every string $x$ viewed as a left node let LIST($x$) be the set of $x$’s right neighbors.

1. If in $G$ the Matcher has a winning strategy with overhead $a(n)$, then the set LIST($x$) translated into standard machine descriptions contains an $a(n) + O(1)$-short program for every $x$.

2. If LIST($x$) contains an $a(n)$-short program for every $x$, then in $G$ the Matcher has a winning strategy with overhead $a(n) + O(1)$.

**Proof of (i).** Run $U$ on all binary strings by dovetailing. When $U(q) = x$ for some $x$ and some $q$, then the Requester makes the request $(x, |q|)$ (provided it was not made earlier). By assumption on $G$, the Matcher effectively matches $x$ to one of its neighbors $y$ of length at most $|q| + a(n)$. Note that there exists a machine which on input $y$ outputs $x$ because the machine can effectively replay this entire process until it witnesses that the Matcher matches $x$ to $y$. Therefore when $|q| = C(x)$, the process above witnesses that LIST($x$), or more precisely a translation of it for the standard machine, contains a description for $x$ of length at most $C(x) + a(n) + O(1)$. The translation adds the $O(1)$ term.

**Proof of (ii).** We argue by contrapositive. Assume that for any computable Matcher strategy and for any $i$, there exists a computable Requester strategy that issues requests which eventually the Matcher fails to satisfy online with overhead $a(n) + i$. For every $i$, we consider the following Matcher$_i$ strategy. When a new request $(x, k)$ is made, Matcher$_i$ executes $U(p)$ in parallel for all $p \in$ LIST($x$) with length $|p| \leq k + a(n) + i$, and if one of these $U(p)$ halts and outputs $x$, he matches $x$ to $p$. As per our assumption, there exists a Requester strategy which issues requests such that the above Matcher, eventually fails. Such a Requester strategy can be found effectively (see [2] for details).

Let $(x, k)$ be an arbitrary request made by the Requester against Matcher$_i$. Note that $x$ can be described by $k, i$, and the ordinal of $(x, k)$ among the at most $2^k$ requests of the form $(\cdot, k)$. We can represent $k$ and the above ordinal using $k$ bits by writing the ordinal on a string of length exactly $k$, and $i$ can be written in a self-delimited way using $2 \log i + 1$ bits. It follows that $C(x) \leq k + 2 \log i + O(1) \leq k + i$, for some large enough $i$, which we now fix. Eventually some request $(x, k)$ sent to Matcher$_i$ is unsatisfiable, but this happens only when LIST($x$) has no program $p$ for $x$ of length bounded by $k + a(n) + i \leq C(x) + a(n)$, i.e., when LIST($x$) does not contain an $a(n)$-short program for $x$.

Next we construct bipartite graphs with small degree $d(n)$ in which the Matcher wins with small overhead $a(n)$. Note that the degree and overhead parameters in our example below are not optimal.
Lemma 7 (based on [2]). There exists a computable bipartite graph $G$ with left side $L = \{0,1\}^*$, degree $d(n) = O(n^3)$ and with a winning Matcher strategy with overhead $a(n) = O(\log n)$. The matching strategy is the greedy one.

Proof. A bipartite graph $G$ is a $(s,s')$-expander if every $s$ left nodes have $s'$ neighbors. The classic Hall’s Marriage Theorem [29] shows that $s$ off-line matching requests are satisfied if and only if for every $s' \leq s$, $G$ is an $(s',s')$-expander. Expander graphs are useful for online matching as well.

Claim. Let $G$ be a $(s,s+1)$-expander. If $2s$ left nodes request online matching, it is possible to satisfy at least $s$ of them.

We satisfy the $2s$ requests in the greedy way: when a left node $x$ makes a request, we match it with the first available neighbor (if one exists). Now, suppose $s$ requests cannot be satisfied. The $s$ rejected left nodes have a total of at least $s+1$ neighbors by the expansion assumption on $G$. Since none of the rejected nodes are available, their neighbors must have been used to satisfy $s+1$ requests. This means that at least $s + (s+1) = 2s+1$ requests have been made, which is a contradiction and proves the claim.

For simplicity of presentation, we assume that in all requests $(x,k)$ it holds that $k \leq |x|$ (rather than $k \leq |x| + O(1)$) and leave the necessary minor adjustments to the motivated reader. Using the probabilistic method and exhaustive search, it can be shown that, for every $n$ and every $k \leq n$, there exists a bipartite graph $G_{n,k}$ that is a $(2^k,2^k+1)$-expander with left side $L_{n,k} = \{0,1\}^n$, right side $R_{n,k} = \{0,1\}^{k+2}$, and left degree $d(n) = n+1$ [2]. Recall that the union of a collection of bipartite graphs is the bipartite graph whose left set, edges, and right set consist of the union of the component left sets, edges, and right sets respectively. We build the graph $H_{n,k}$ by taking the union $G_{n,k-1} \cup \cdots \cup G_{n,1}$ as illustrated in the figure below.

Figure 2: Graph $H_{n,k}$ is the union of $G_{n,1} \cup \cdots \cup G_{n,k-1}$ and satisfies all requests $(x,k)$ with $|x| = n$.

Now consider matching requests in $H_{n,k}$ of the form $(x,k)$ with $|x| = n$. By the definition of the Matcher-Requester game, there are at most $2^k$ such requests. It follows from the Claim above that more than half of these requests are satisfied with right nodes in $R_{n,k-1}$, half of the remaining ones are satisfied in $R_{n,k-2}$, and so on. Therefore all such requests are satisfied in $H_{n,k}$ with overhead 1 as the length of right nodes is bounded by $k+1$. Note that $H_{n,k}$ has degree $(k-1)(n+1) < n^2$. 

\[
L_{n,1} = \ldots = L_{n,k-1} = \{0,1\}^n
\]

\[
R_{n,k-1} = \{0,1\}^{k+1} < 2^k/2 \text{ rejected requests}
\]

\[
R_{n,k-2} = \{0,1\}^k < 2^k/4 \text{ rejected requests}
\]

\[
R_{n,1} = \{0,1\}^3 < 1 \text{ rejected requests}
\]
Next we take a second union \( H_n = H_{n,n} \cup \cdots \cup H_{n,1} \) where we append to each right node in \( H_n \) a self-delimiting code of length \( O(\log n) \) so that for the right side of the bipartite graphs we have a disjoint union. The graph \( H_n \) satisfies with overhead \( O(\log n) \) all requests of the form \((x,k)\) with \(|x| = n\) and has degree less than \( n^3 \).

Finally, we define \( H \) as the union of \( H_1, H_2, \ldots \), where again we add \( O(\log n) \) bits to the right nodes in each \( H_n \) to obtain a disjoint union. This graph has degree bounded by \( n^3 \) and satisfies all requests with overhead \( O(\log n) \).

With more work, the construction in Lemma 7 can be refined to have overhead \( O(1) \) and degree \( O(n^2) \). Combining this improvement with Lemma 6 yields the Optimal-Length Short List Theorem. Using a derandomization technique, one can obtain the Polynomial-Time Short List Theorem, and we sketch a proof of the necessary construction in the addendum to this work.

**Remark.** One can view the requests in Lemma 7 as a continuous process in which left nodes first request and later release right nodes. Suppose that the left nodes in Lemma 7 (see Figure 2) are women and the right nodes are dance instructors at a ballroom. Each woman may request a dance with one of her right neighbors with whom she has practiced the rhumba before. Assuming that at any moment the women make at most \( 2^k \) dance requests in total, the dance instructors can accommodate all of them. This Latin dancing example, which personifies a rather generic client-server scheme, may lend itself to further applications; note that the Explicit Online Matching Theorem from the previous section offers an efficient way to generate the necessary underlying bipartite graph.

## 4 Short descriptions in networks

We now turn to the case of short descriptions for two or more correlated strings situated at different locations. This problem, generically known as distributed compression, is typically analyzed in settings in which one or more senders transmit data to one or more receivers, but we restrict our discussion to the case of a single receiver. The senders can either share their data and compress it together, or they can compress it separately. What are the possible compression rates in the two cases? Under the assumption that the parties know the complexity profile of the inputs, we will see, counterintuitively, that essentially there is no difference between the two scenarios.

Before we begin, we need to clarify one issue. Whenever we analyze compression, we need to also consider its counterpart, decompression. We have seen that, modulo a few help bits, compression to almost minimum description length can be done in polynomial time (see Theorem 2 and Theorem 4 for the exact statements). On the other hand, decompression cannot be even remotely efficient when compression is done at this level of optimality. This is due to the fact that, for any computable time bound \( t \) and every length \( n \), there exist \( n \)-bit strings, the so-called deep strings [5], which require \( t(n) \) time to reconstruct from any short description.

So, let’s get started\(^{21}\). Suppose that Alice and Zack are collaborating on a document, and Alice wants to update Zack with her changes over a channel with limited bandwidth.\(^ {22}\) If Alice knows which version of the file Zack has, she can send a diff with her changes, and Zack can use this information to synchronize. But suppose Alice has no idea what version of the file Zack

---

\(^{21}\)In our discussion, we use \( C(x \mid y) \), the conditional Kolmogorov complexity of \( x \) given \( y \) (see Section 3), and \( C(x, y) \), the Kolmogorov complexity of a fixed pairing of \( x \) and \( y \).

\(^{22}\)This example is taken from [57, Section 2.2].
has. Can she still concisely communicate the updated file? The following result, which follows from a relativization of the polynomial-time short list construction and improves a result from [45], provides an answer to this question.

**Improved Muchnik’s Conditional Complexity Theorem (66).** For any strings $x$ and $y$, there exists a string $p$ such that

1. $|p| = C(x|y)$,
2. one can compute $x$ from $p$ and $y$ using $O(1)$ help bits, and
3. one can efficiently compute $p$ from $x$ using $O(\log |x|)$ help bits.

The hidden constants do not depend on $x$, $y$, or $p$.

Thus, Alice can efficiently compress her file $x$ to minimal description length without knowing Zack’s version $y$ (item (1) and item (iii)), although they need a few help bits for compression (item (iii)) and decompression (item (ii)). Musatov, Romashchenko and Shen [47], in the paper that has introduced online matchings in Kolmogorov complexity theory, show a version of Muchnik’s Theorem for space bounded Kolmogorov complexity, in which both compression and decompression are space-efficient.

Muchnik’s Theorem is just one particular instance of a more general phenomenon. To present it, we recall the example in the introduction in which Alice knows a line $\ell$ in the affine plane over $F_2^n$, Bob knows a point $P$ on $\ell$, and they want to send $\ell$ and $P$ to Zack. In a more abstract setting, Alice and Bob have correlated $n$-bit long strings $x$ and respectively $y$, which they want to send to Zack. How many bits do they need to send, or, in other words, what are the possible compression rates? Under the name *lossless distributed compression*, this problem has been investigated in information theory. In this framework, $x$ and $y$ are realizations of random variables $X$ and $Y$. The standard stance in information theory is to assume that the pair $(X,Y)$ is 2-DMS (2-discrete memoryless sources), which, in the binary case, means that $(X,Y)$ consists of $n$ independent drawings of pairs of random bits from a single joint distribution $p(b_1, b_2)$. We want to see for what integers $n_1$ and $n_2$, there exist encoding functions $E_i : \{0,1\}^n \rightarrow \{0,1\}^{n_i}, i = 1, 2$, and decoding function $D$, such that $D(E_1(X), E_2(Y)) = (X,Y)$ with high probability. It is not difficult to see that it is necessary that $n_1 + n_2 \geq H(X,Y)$, $n_1 \geq H(X|Y)$, and $n_2 \geq H(Y|X)$, where $H$ is the Shannon entropy function. The classic Slepian-Wolf Theorem [58] roughly speaking states that these inequalities are also sufficient and thus completely characterizes the compression rates for correlated sources in the case of 2-DMS (see [58] or [12] for the exact statement).

Similarly to Muchnik’s Theorem, it is striking that Alice and Bob can compress separately and achieve optimal compression rates. For example, suppose $X$ and $Y$ have information measures similar to $\ell$ and $P$, that is $H(X) = 2n$, $H(Y) = 2n$, and $H(X,Y) = 3n$. Then $(n_1 = 1.5n, n_2 = 1.5n)$ satisfy the inequalities in the Slepian-Wolf Theorem and therefore, assuming $(X,Y)$ is 2-DMS, they are achievable compression rates. This means that Alice can compress her $2n$-bit realization of $X$ to $1.5n$ bit without knowing the other sender’s string, and Bob can do the same. They cannot beat this even if they share $X$ and $Y$! Thus, distributed compression is as good as centralized compression in the case of 2-DMS. However, the 2-DMS model is rather restrictive because of the independence condition. The geometrical correlation between $\ell$ and $P$, simple as it is, is not captured by the 2-DMS model. Perhaps more importantly, in real-world applications, $X$ and $Y$ correspond typically to successive iterations of a stochastic process and usually these iterations are not independent.
These shortcomings can sometimes be surmounted using Kolmogorov complexity, abbreviated KC in the rest of this section. For example, the correlation between \( \ell \) and \( P \) can be stated as \( C(\ell \mid P) \leq n + O(1) \). Romashchenko [53] has obtained a KC-analogue of the Slepian-Wolf theorem. His result is valid for any constant number of senders (the same holds for the Slepian-Wolf Theorem), but, for simplicity, we present it for the case of two senders: For any two \( n \)-bit strings \( x \) and \( y \) and any two numbers \( n_1 \) and \( n_2 \) such that \( n_1 \geq C(x \mid y) \), \( n_2 \geq C(y \mid x) \) and \( n_1 + n_2 \geq C(x, y) \), there exist two strings \( p_1 \) and \( p_2 \) such that \( |p_1| = n_1 + O(\log n) \), \( |p_2| = n_2 + O(\log n) \), \( C(p_1 \mid x) = O(\log n) \), \( C(p_2 \mid y) = O(\log n) \) and \( C(x, y \mid p_1, p_2) = O(\log n) \). In words, for any \( n_1 \) and \( n_2 \) satisfying the necessary conditions, Alice can compress \( x \) to a string \( p_1 \) of length just slightly larger than \( n_1 \), and Bob can compress \( y \) to a string \( p_2 \) of length just slightly larger than \( n_2 \) such that Zack can reconstruct \((x, y)\) from \((p_1, p_2)\), provided all the parties use a few help bits.

Recently, [73] has established a KC-analog of the Slepian-Wolf theorem in which the help bits are replaced by the complexity profile of the input. The result holds for any constant number of senders, but for simplicity we present it for the case of two senders. In this case, the assumption is that Zack knows the complexity profile of \( x \) and \( y \), which is the tuple \((C(x), C(y), C(x, y))\) (note that in the classic Slepian-Wolf theorem, the parties also need to know the information-theoretical profile of the sources in terms of Shannon entropy). Alice is using a probabilistic compressor \( E_1 \) that compresses from length \( n \) to length \( n_1 \), Bob is using a probabilistic compressor \( E_2 \) that compresses from length \( n \) to length \( n_2 \), and Zack is using decompressor \( D \) such that for all \((x, y)\), \( D \) on input \((E_1(x), E_2(y))\) and the complexity profile of \( x \) and \( y \), returns \((x, y)\) with high probability. The question is for what values of \( n_1 \) and \( n_2 \) is this possible. It is easy to see that, assuming the error probability is bounded away from 1, it is necessary that \( n_1 + n_2 \geq C(x, y) - O(1) \), \( n_1 \geq C(x \mid y) - O(1) \), and \( n_2 \geq C(y \mid x) - O(1) \). The following theorem shows that essentially the above conditions are also sufficient (modulo a small polylogarithmic overhead), provided Zack knows the complexity profile of \( x \) and \( y \).

**Theorem 8 (KC Slepian-Wolf Theorem [73]).** There exist probabilistic polynomial-time algorithms \( E_1, E_2 \) and algorithm \( D \) with the following property: For every \( n \), for every \( n_1, n_2 \) and for every \( n \)-bit strings \( x \) and \( y \), if \( n_1 + n_2 \geq C(x, y) \), \( n_1 \geq C(x \mid y) \) and \( n_2 \geq C(y \mid x) \), then

1. \( E_1 \) on input \( x \) and \( n_1 \) returns a string \( p_1 \) of length bounded by \( n_1 + O(\log^3(n/\epsilon)) \), \( E_2 \) on input \( y \) and \( n_2 \) returns a string \( p_2 \) of length bounded by \( n_2 + O(\log^3(n/\epsilon)) \), and

2. \( D \) on input \((p_1, p_2)\) and the complexity profile of \( x \) and \( y \) returns \((x, y)\) with probability \( 1 - \epsilon \).

The decompressor \( D \) does not run in polynomial time for the same reason we have seen above. Remarkably, Chumbalov and Romashchenko [10] have obtained Slepian-Wolf coding for two strings with randomized polynomial time compression and decompression in the so-called combinatorial case, in which correlation is measured by Hamming distance.

Strictly speaking, the KC Slepian-Wolf Theorem [8] and the classic Slepian-Wolf Theorem are incomparable because they assume different models (however, the classic result can be obtained from Theorem 8 taking into account the well-known relation between Kolmogorov complexity and Shannon entropy for discrete memoryless sources, see [12, Ch.14] or [57, Ch.7]). Theorem 8 is applicable for sources that are algorithmically correlated such as \( \ell \) and \( P \) in our geometrical example. It shows that for such sources, for any compression rates that satisfy the obvious necessary conditions, distributed compression is on a par with centralized compression. This is perhaps even more surprising than in the classic Slepian-Wolf theorem, because in Theorem 8 there is no type of independence assumption (note that, in principle, independence helps distributed compression; after
all, in the limit case of full independence, it makes no difference whether compression is distributed or centralized).

One final remark. The Slepian-Wolf Theorem has been the seminal result that spawned network information theory. While the information-theoretical side of this field is well-established and dynamic (see the recent monograph [18], and the dedicated chapters in [12], [13]), a lot of work remains to be done on the KC side (see [57, Chapter 12]).

References


A Technical vignettes

We now present proofs of a few theorems discussed in this paper. The following is Stephan’s unpublished proof of Kinber’s result from [34].

Theorem 9. There exist acceptable numberings \( \varphi \) and \( \psi \) such that \( \text{MIN}_\varphi \) is btt-incomparable to \( \text{MIN}_\psi \).

Proof. Fix a canonical enumeration of the binary strings (in order of nondecreasing length), and identify the \( n \)th natural number with the \( n \)th string in this enumeration. Let \( I_0 \) be the interval consisting of binary strings of length 1 and inductively, when \( I_0 \cup \cdots \cup I_k \) contains all strings up to length \( k \), define \( I_{n+1} \) to contain all strings of lengths \( k+1 \) through \( 2^k \) inclusive. Let \( \nu \) be any acceptable numbering.

We will build acceptable numberings \( \varphi \) and \( \psi \) by coding the function \( \nu_e \) into interval \( I_{2e} \) for \( \varphi \) and into interval \( I_{2e+1} \) for \( \psi \). All \( \varphi \) and \( \psi \) indices not involved in this coding will compute the everywhere undefined function. Suppose the longest strings in \( I_{2e} \) have length \( \ell \), and let \( x \) be the canonically greatest string of length \( \ell \) satisfying \( C(x) \geq \ell \). Such a string \( x \) exists simply by scarcity of shorter descriptions. We construct our numbering \( \varphi \) so that every \( \varphi \)-index in \( I_{2e} \) greater than or equal to \( x \) computes \( \nu_e \), and every other \( \varphi \)-index in the interval is undefined. Define \( \psi \) in an analogous way on \( I_{2e+1} \). The numberings \( \varphi \) and \( \psi \) are both acceptable because for all \( e \), \( \varphi_{\max}(I_{2e}) = \psi_{\max}(I_{2e+1}) = \nu_e \), and note that all elements \( \text{MIN}_\varphi \) and \( \text{MIN}_\psi \) occur only at elements of the special form \( x \in \text{RAND} \) above.

In any given btt-reduction \( f \) witnessing \( \text{MIN}_\psi \leq_{\text{btt}} \text{MIN}_\varphi \) and for all but finitely many \( \psi \)-indices \( z \) which do not compute the everywhere undefined function, \( f \) on input \( z \) does not query any elements in \( \text{MIN}_\varphi \) greater than \( z \). Indeed, if \( f \) were to query such an element, we would have for all but finitely many \( k \) a method for computing a string with complexity greater than \( 2^k \) using at most \( k + O(1) \) bits.

Under the assumption that \( \text{MIN}_\psi \leq_{\text{btt}} \text{MIN}_\varphi \), we now derive that members of RAND exist with short descriptions. Using brute force search, one can describe \( x \in \text{MIN}_\varphi \) from the set of bounded truth-table queries for \( x \) whenever \( x \) is the least number to use these particular queries. One can even describe this \( x \) without knowing the queries which belong to the same interval as \( x \) since by construction all such indices are not \( \varphi \)-minimal. Thus we need at most \( ck + O(1) \) bits to describe \( x \), where \( k+1 \) is the length of smallest index in the interval containing \( x \) and \( c \) is the fixed number of queries for the given bounded-truth-table reduction. Since \( C(x) \geq 2^k \), we reach a contradiction if there exist infinitely many such \( x \), and there are: if all but finitely many \( \varphi \)-minimal indices were to make similar truth-table queries, then \( \text{MIN}_\varphi \) would be computable. A symmetric argument shows that \( \text{MIN}_\varphi \not\leq_{\text{btt}} \text{MIN}_\psi \).

Using the Kummer Cardinality Theorem, we derive a solution to Schaefer’s MIN∗ Problem for the case of numberings with the Kolmogorov property.

Theorem 10 ([32]). For any \( \psi \) with the Kolmogorov property, \( \text{MIN}^*_\psi \geq_T K \).
Proof. Let $\sigma_n$ be the $n^{th}$ finite string in an enumeration of $\mathbb{N}^*$. Using the Kolmogorov property, which guarantees that $\text{MIN}^*_\psi$ is not hyperimmune, partition $\mathbb{N}$ into computable intervals $\{I_n\}$ such that for all $n$ there exists a $z \in \text{MIN}^*_\psi$ such that $\min(I_n) \cdot (|\sigma_n| + 1) + |\sigma_n| < z < \max(I_n)$. Define (via a stagewise construction) a numbering $\nu$ such that for all $p \in I_n$,

$$\nu_p = \psi_p((|\sigma_n|+1)+[K(a_1)+K(a_2)+...+K(a_{|\sigma_n|})])$$

(A.1)

where $\sigma_n = a_1a_2...a_{|\sigma_n|}$. Also, define $\text{MIN}^*_\psi$-computable functions $f$ and $g$ satisfying $f(n) \cdot (|\sigma_n| + 1) + g(n) = \max(I_n \cap \text{MIN}^*_\psi)$, and $g(n) \leq |\sigma_n|$.

By the Kolmogorov property, there exists a constant $c$ satisfying

$$(\forall p) (\exists e < pc + c) [\psi_c = \nu_p].$$

(A.2)

Let $\sigma_m = b_1b_2...b_k$ be any length $k$ string satisfying $k \geq c$ and $\min(I_m) \geq c$. Since $f(m) \in I_m$, we have $f(m) \geq c$, and by definition of $f$ and $g$,

$$f(m) \cdot (k + 1) + g(m) \in \text{MIN}^*_\psi.$$

(A.3)

It follows from (A.2) that some $\psi$-index less than $f(m)c + c$ computes $\nu_f(m)$, so by (A.1) and the fact that $f(m) \cdot (k + 1) \geq f(m)c + c$ we obtain

$$f(m) \cdot (k + 1) + [K(b_1) + ... + K(b_k)] \notin \text{MIN}^*_\psi.$$

(A.4)

Combining (A.3) and (A.4) yields $g(m) \neq K(b_1) + ... + K(b_k)$. Now using the fact that membership in $K$ for strings $\sigma_n$ with either $|\sigma_n| < c$ or $\min(I_n) < c$ can be decided computably, it follows from the Kummer Cardinality Theorem (relativized to $\text{MIN}^*_\psi$) that $K \leq_T \text{MIN}^*_\psi$. 

Next is our rendition of Bauwens and Shen’s proof [1] of the Improved Gács Theorem from Section 3.

**Theorem 11.** For every $n$, there is a string $x$ of length $n$ such that $C(C(x) \mid x) \geq \log n - O(1)$.

**Proof.** The proof is by diagonalization, but its combinatorial mechanism can be best understood in terms of a game between two parties, White and Black. In fact, Black’s moves are obvious to White’s actions as they are determined solely by the universal machine, and thus, we have a game between one player (White) and “Nature,” where “Nature” means here the universal machine. White is trying to find a pair $(i, x)$ such that (1) $C(i \mid x) \geq \log |x| - 1$, and (2) $C(x) > i$. When Black discovers violations of (1) or (2), he kills the White’s attempt, who is forced to try a new $(i, x)$. In the end, we show that White succeeds to have one attempt $(i, x)$ to stay alive forever, and moreover $x$ in this surviving attempt satisfies (3) $C(x) \leq i + O(1)$. Note that (2) and (3) imply $C(x) = i + O(1)$ and now (1) becomes $C(C(x) \mid x) \geq \log |x| - O(1)$, as desired.

The game is played on infinitely many boards $B_n$, one for every $n$. The board $B_n$ has $n$ rows and $2^n$ columns. The rows are labeled $n - 1, n - 2, \ldots, 0$ (in this order from top to bottom), and the columns are labeled with $n$-bit long binary strings in alphabetical order. The cell $(i, x)$ is the cell on row $i$ and column $x$. Both players will put tokens of their color on the cells of the boards, and in addition Black can also mark some cells.

Black plays as follows. He enumerates in a dovetailing manner computations of the universal machine $U$ of the form $U(p, x)$ and $U(p)$ for all $p$ and $x$, in the process determining all pairs $(i, x)$
such that (a) $U(p, x) = i$ for some $p$ of length less than $\log |x| - 1$ or (b) $U(p) = x$ for some $p$ of length less than $i$. When Black discovers a pair $(i, x)$ such that $C(i \mid x) < \log |x| - 1$ (case (a)), he marks cell $(i, x)$ on board $B_{|x|}$. Note that he marks less than $|x|/2$ cells in column $x$. When Black discovers that $C(x) \leq i$ (case (b)), he puts a black token on cell $(i, x)$ on board $B_{|x|}$. A total of at most $2^i - 1$ black tokens can be put on row $i$ on all boards.

White plays as follows. At some moments she puts white tokens on the boards. A white token placed on cell $(i, x)$ is killed if either Black marks the cell $(i, x)$, or if Black puts a black token in the same column but below it, that is on a cell $(i', x)$ with $i' \leq i$. White starts by putting a token on the cell in the north-west corner of every board $B_n$. Then she waits till one of her tokens, say the one on cell $(i, x)$, is killed. If her white token has been killed by Black marking the cell, then White puts a new white token on the closest cell below it that is not marked. Since Black marks less than $|x|/2$ cells on column $x$, and the column has $|x|$ cells on column $x$, White can always move. If on the other hand, the white token has been killed by Black putting a black token below it, then White chooses the first column on the same board that has no black tokens. Since Black can put at most $\sum_{i=1}^{n-1}(2^i - 1) < 2^n$ tokens on $B_n$ (because there can be at most $2^i - 1$ black tokens on row $i$), there is always a column without any black tokens. Next White puts a token on the top-most cell of this column that is not marked. By the same argument as before, there are unmarked cells on that column (actually more than half of the cells are not marked). So, White can move in this case as well. In both cases, the white token is put somewhere in the first $|x|/2$ rows. It is important to note that White's moves can be effectively enumerated.

Since the number of White and Black moves on each board is finite, it follows that each board $B_n$ eventually stabilizes and there is one white token on a cell $(i, x)$ that remains alive forever. We call this $(i, x)$ the winning cell for $B_n$. Let us analyze such a winning cell $(i, x)$. We show below that the total number of white tokens in row $i$ (on all boards) is $O(2^i)$. This implies that $C(x) \leq i + O(1)$. To see this, note that $x$ can be described by $i$ and the rank of $(i, x)$ among the at most $O(2^i)$ cells with white tokens on them in an enumeration of these cells in the order in which the white tokens are put. Moreover, this description of $x$ can be written on $i + O(1)$ bits. It also holds that $C(x) \geq i$ (otherwise Black would put a black token on cell $(i, x)$). Combining the two inequalities, it follows that $i = C(x) + O(1)$. We also know that $C(i \mid x) \geq \log |x| - 1$ (otherwise Black would mark cell $(i, x)$). Therefore, we obtain $C(C(x) \mid x) \geq \log |x| - O(1)$.

It remains to prove our claim that the total number of white tokens on row $i$ on all boards is bounded by $O(2^i)$. We count separately the number of dead tokens and the number of living tokens (after the boards have stabilized). Each dead white token (except those in the same column with the winning cell) has a black token below it. So the total number of dead white tokens on row $i$ is bounded by $\sum_{j=0}^{i-1} 2^j < 2^i$ (we took into account that there are at most $2^i$ black tokens on row $j$ on all boards). Next we bound the number of living white tokens on row $i$. No such token exists on any board $B_n$ with $n \geq 2i$, because on $B_n$ no white token is placed below row $n/2$. Also, recall that there exists a single living white token on each board. Therefore, the total number of living white tokens on row $i$ is at most $2i$. Our claim and the theorem are proved.

The following is the polynomial counterpart of Lemma 7. In combination with Lemma 6, it yields Theorem 2. We recall that a graph is explicit if given $x$ and $i$, the $i$-th neighbor of $x$ can be computed in polynomial time.

**Lemma 12** (based on [66, 72]). For every $\epsilon > 0$, there exists an explicit bipartite graph $G$ with left side $L = \{0, 1\}^*$, degree $d(n) = \text{poly}(n)$ and with a winning Matcher strategy with overhead
Proof sketch. At the core of the construction in the proof of Lemma 7 is the \((2^k, 2^k + 1)\)-expander \(G_{n, k}\), obtained with the probabilistic method. Essentially, we need to replace it with an explicit graph. Looking ahead, it helps if we get an explicit \((2^k, 2^k + 1)\)-expander (thus with expansion by a multiplicative factor of 2).

We use the bipartite expander construction of Guruswami, Umans, and Vadhan \[28\] and obtain, for every \(n\) and every \(\log n \leq k \leq \frac{n}{2}\), a bipartite graph \(G_{n, k}\) with degree \(D = \text{poly}(n)\), left side \(L = \{0, 1\}^n\), right side \(R = \{0, 1\}^{O(k)}\) such that for any \(K' \leq 2^k\), any subset of \(K' \leq 2^k\) left nodes has at least \(2K'\) neighbors. Thus we have the desired factor-2 expansion for any left set of size at most \(2^k\).

Let us fix \(k\). We first build a graph that satisfies all requests of the form \((x, k)\). For simplicity we will assume in this sketch that in all these requests it holds that \(|x| \leq k\) (instead of \(k + O(1)\)), and also \(k \geq \log |x|\) (the case of smaller \(k\) can be handled in a simple “brute-force” manner, whose details we omit). First we take \(H_k = G_{n, k} \cup G_{n, k+1} \cup \ldots \cup G_{2k, k}\) (with disjoint union as we did in Lemma 7). The graph \(H_k\) is a \((2^k, 2^k + 1)\)-expander. This is so because any set of left nodes of size \(2^k\) can be partitioned by putting all strings of the same length in a subset of the partition, and each subset of the partition expands by a factor of 2, because it has size \(K' \leq 2^k\) and its elements live in one of the graphs in the disjoint union.

We have almost what we want, except that the right side of \(H_k\) has size \(2^{O(k)}\) instead of \(2^{k+O(1)}\), which is needed for matching with overhead \(O(1)\). This is solved via disperser graphs. A \((2^k, \delta)\)-disperser graph \(G = (L, R, E \subseteq L \times R)\) is a \((2^k, (1 - \delta)|R|)\) expander.

We build \(G_k\) as the composition of \(H_k\) with an explicit disperser \(TUZ_k\) of Ta-Shma, Umans, and Zuckerman \[64\] (“composing” means that \((x, z)\) is an edge in \(G_k\) if there exists \(y\) such \((x, y)\) is an edge in \(H_k\) and \((y, z)\) is an edge in \(TUZ_k\)). By choosing the appropriate parameters, \(TUZ_k\) has polynomial degree, its left side is equal to the right side of \(H_k\), its right side has size \(2^{k+O(1)}\), and each set of \(2^{k+1}\) left nodes in \(TUZ_k\) has \(2^{k+1}\) neighbors. So, the factor-2 expansion holds in \(G_k = H_k \circ TUZ_k\), because each set of left nodes of size \(2^k\) expands by a factor of 2 when we make the left-to-right transition in \(H_k\) and then it preserves its size in the second left-to-right transition in \(TUZ_k\). Note that \(G_k\) has polynomial degree and its right nodes have length \(k + O(1)\). As it is now, by the claim used in the proof of Lemma 7 \(G_k\) satisfies at least half of all requests of the form \((x, k)\), for all \(x\) with \(|x| \leq 2^k\). After a further modification similar to the one in Figure 2 it can be shown as we did in Lemma 7 that the modified \(G_k\) satisfies all the matching requests of the above type. It follows that \(G_k\) can satisfy all requests of the form \((x, k)\) with \(O(1)\) overhead.

Next we simply take the union \(\bigcup_k G_k\) to handle all \(k\)’s. Note that a string \(x\) appears as a left node in \(|x| + O(1)\) graphs \(G_k\), so the degree of the union graph is still polynomial. 

\(a(n) = O(1)\). The matching strategy is the greedy one.