Sparse Sets versus Complexity Classes

Jin-Yi Cai
Mitsunori Ogihara

ABSTRACT The study of sparse hard sets and sparse complete sets has been a central research area in complexity theory for nearly two decades. Recently new results using unexpected techniques have been obtained. They provide new and easier proofs of old theorems, proofs of new theorems that unify previously known results, resolutions of old conjectures, and connections to the fascinating world of randomization and derandomization. In this article we give an exposition of this vibrant research area.

1 Introduction

Complexity theory is concerned with the quantitative limitation and power of computation. During the past several decades computational complexity theory developed gradually from its initial awakening [Rab59, Yam62, HS65, Cob65] to the current edifice of a scientific discipline that is rich in beautiful results, powerful techniques, fascinating research topics and conjectures, deep connections to other mathematical subjects, and of critical importance to everyday computing.

The building blocks of complexity theory are the various complexity classes. A complexity class consists of computational problems that are classified according to their inherent computational costs to solve the problems. There are various complexity measures, the most basic of which are cost in time and space. Complexity theory is concerned with various measures of complexity and their inter-relationships, such as time and space, determinism and non-determinism, sequential versus parallel computation, and deterministic versus probabilistic computation.

The most well known complexity classes are the space bounded classes PSPACE, DSPACE[$\log^{O(1)} n$], LOGSPACE; time bounded classes EXP, E, P; nondeterministic time classes NEXP, NE, NP; parallel complexity classes NC^1, NC^2, etc; probabilistic classes ZPP, RP, BPP, PP; counting classes #P, $\oplus$P; and the polynomial time hierarchy PH defined in terms of alternations Δ_i^p, Σ_i^p, Π_i^p. We assume the reader is familiar with most of these classes. We will also be concerned with various reducibilities $\leq_r^{\mathcal{C}}$, where $\mathcal{C}$ is a complexity class, denoting the power available to the reduction such as

P or L (for LOGSPACE), and where r denotes a type of reducibility such as many-one, or Turing, or bounded truth-table reducibilities. The power available to the reduction should be somewhat weaker than the power of the class for which the reduction is applied. For instance, within NP, we can consider reducibilities $\leq_m^P$, $\leq_T^P$ or $\leq_{btt}^P$. Within P, we can consider $\leq_m^L$. We refer the readers to any standard textbook on definitions of complexity classes and reducibilities mentioned here, for example [BDG88].

The complexity classes mentioned above are all based on *uniform* complexity measures, where a Turing machine which has a code of a fixed length will handle all inputs of increasing length. This is the usual concept of an algorithm which solves a computational problem consisting of infinitely many instances. Contrary to that, there is the important notion of *non-uniform* complexity, where, for instance, a boolean circuit of certain size exists that solves a problem for all instances of size n. The asymptotic growth of the size of the minimal circuit for all instances of size n can be taken as a non-uniform measure of complexity of the problem. It is non-uniform because no uniform procedure is given to find such a circuit for each length n.

Sparse sets are intimately related to non-uniform complexity.

A sparse set is a set with only a small number of strings per length. More precisely, a sparse set has at most $p(n)$, a polynomial number of strings up to length n. Thus such a set can only encode a small amount of information; it is thought of as languages with small description. Sparse sets have played an important role in complexity theory; above all, with respect to various reducibilities, we ask:

Does a complexity class $\mathcal{C}$ have a sparse hard or complete set?
What consequences follow from such an assumption?

Generally speaking, as sparse sets are sets of low information content, to have a sparse hard or complete set places severe limitations on a complexity class. Alternatively, having a sparse hard or complete set can be viewed as having a strong handle on the power of the complexity class. Many consequences are known assuming the existence of sparse hard sets. For a survey of earlier results, see [Mah89, HOW92, You92].

The two questions raised here are fascinating because they link together both uniform complexity, in terms of a complexity class and a reducibility, and non-uniform complexity, via a sparse set. The $\leq_T^P$-closure of sparse sets is known [BDG88] to be identical to the class of languages having polynomial-size circuits, and to P/*poly*—the class of languages decidable in polynomial time with short advice, introduced by Karp and Lipton [KL82]. The connection to non-uniform complexity, especially polynomial size circuits, was the first primary motivation for the study of sparse sets. A fundamental result of Karp and Lipton [KL82] states that if a sparse set is NP-hard under polynomial time Turing reduction $\leq_T^P$, i.e., every language in NP has polynomial size circuits, then the polynomial time hierarchy collapses to its second level.

The second major motivation for the study on sparse sets was the isomorphism conjectures of Berman-Hartmanis for NP [BH77], and the conjectures for P by Hartmanis [Har78].

In 1976, L. Berman and Hartmanis [BH77] showed that all NP-complete languages known at the time (such as those found in [GJ79]) under $\leq_m^P$-reductions are isomorphic under one-to-one, onto, P-computable, and P-invertible functions on Σ^*. They went on to conjecture that this is true for all NP-complete languages under $\leq_m^P$. As typical NP-complete languages such as SAT are all exponentially dense, it follows from the isomorphism conjecture that no sparse set can be NP-complete under $\leq_m^P$. This becomes the sparse non-completeness conjecture for NP by Berman and Hartmanis. This latter conjecture was resolved by Mahaney, who showed that this is true if and only if NP $\neq$ P [Mah82].

While Karp-Lipton's theorem deals with $\leq_T^P$, and Mahaney's theorem settles the case for $\leq_m^P$, the case for polynomial time bounded truth-table reducibility $\leq_{btt}^P$, which is intermediate between $\leq_m^P$ and $\leq_T^P$, was left open for ten years. It was settled by Ogihara and Watanabe [OW91] in 1990. They showed that the existence of sparse $\leq_{btt}^P$-hard sets for NP implies P = NP. This is a remarkable achievement, unifying many previous partial results. Their proof introduced the notion of a *left-set* which is very effective. We will demonstrate its usefulness by giving easier proofs of earlier results of Mahaney [Mah82] and Fortune [For79].

The issue of isomorphism for complete languages in P was also raised by Hartmanis in [Har78]. Just as in the case of NP, he showed that all known P-complete languages under logspace computable many-one reduction $\leq_m^L$ are isomorphic under one-to-one, onto, logspace computable, and logspace invertible functions on Σ^*. Hartmanis then conjectured that this is true for all P-complete languages under $\leq_m^L$-reductions. Again typical P-complete languages such as CVP, the circuit value problem [Lad75], are all exponentially dense, it follows that no sparse set can be P-complete under $\leq_m^L$. This is the Hartmanis conjecture on the non-existence of sparse complete sets in P. A similar conjecture was also made by Hartmanis [Har78] for NL, the nondeterministic logspace.

However, unlike in the NP case, for nearly two decades, Hartmanis's conjecture for P remained open with little progress until very recently. In a breakthrough, Ogihara [Ogi95] showed that if sparse $\leq_m^L$-hard sets for P exist, then P is contained in poly-logarithmic space DSPACE[$\log^2 n$]. In order to prove his theorem, Ogihara introduced an ingenious auxiliary language based on the P-complete language CVP, which has a certain built-in self-reducibility. His proof then proceeds by solving CVP in DSPACE[$\log^2 n$] with a clever searching procedure using the P-hard sparse set.

This was improved by Cai and Sivakumar [CS95a]. The starting point of their proof is Ogihara's auxiliary language. Their proof consists of three stages. First they showed that under the hypothesis that a sparse hard set exists for P under $\leq_m^L$, then P can be simulated in randomized NC^2. It is interesting to note that the original problem speaks neither parallel nor probabilistic computations. Second with derandomization techniques using finite fields, they showed that the result can be improved to deterministic NC^2. As $NC^2 \subseteq DSPACE[\log^2 n]$, this gives another proof of Ogihara's theorem [Ogi95]. It should be noted that at this stage, there is still an underlying randomized computation to be "simulated" deterministically. Finally, relying more on algebraic techniques, they improved the final result from NC^2 to NC^1, thus resolving Hartmanis's conjecture: There are no sparse complete sets in P under $\leq_m^L$ unless P = LOGSPACE. The Hartmanis conjecture for NL was also resolved: There are no sparse complete sets in NL under $\leq_m^L$ unless NL = LOGSPACE [CS95b].

The Cai-Sivakumar proof is highly algebraic. Under the hypothesis that a sparse hard set exists for P, it is shown that the P-complete circuit value problem CVP can be solved in NC^1, modulo the reduction. As $NC^1 \subseteq$ LOGSPACE, this shows that CVP can be solved in LOGSPACE. However, the proof goes through computations in NC^1—it does not provide a step by step automata theoretic "simulation" of P in LOGSPACE. It is instructive to note that many concepts crucial in this proof, such as NC^1, NC^2, algebraic computations over finite fields, and various randomization and derandomization techniques, were only studied long after, and thus not available at, the time the conjecture was proposed. It is a forceful demonstration of the power of randomization, a point on which we will elaborate further in Section 6. With respect to the philosophical point raised there, namely whether randomization is truly necessary to computation of deterministic problems, it is amusing to note that in the final stage of the Cai-Sivakumar proof all traces of randomization disappeared: there is no underlying randomized computation to be simulated. Randomization only makes its presence felt in its former shadow: one probably would never have arrived at the final proof had it not been for the randomization and derandomization in the earlier stages.

In this article, we give a selective exposition of reseach in this area. In addition to some earlier important results, we focus on some recent technical and conceptual advances that have brought significant achievements in the area. The choices made are subjective ones, and reflect the personal tastes by the authors. Due to page limitations, many fine results in the area are not discussed. The literature on this subject is vast, we refer readers to the bibliography for additional research not covered here.

2 Earlier results for Turing reductions

2.1 Sparse sets and polynomial size circuits

We will start with a theorem attributed to A. Meyer [BH77], see also [BDG88, page 128].

Theorem 2.1 *A set is in the closure of a sparse set under polynomial time Turing reducibility $\le_T^P$ if and only if A has polynomial size circuits.*

Thus, for instance, the proposition that SAT has polynomial size circuits is identical to the assertion that NP is polynomial time Turing reducible to a sparse set S.

The proof of this equivalence is rather easy however, and we will sketch the proof idea here: Suppose A has a family of Boolean circuits $\{C_n\}$, where size $|C_n| \le p(n)$ for some polynomial p, and C_n determines the section $A^{=n} = \{x \in A \mid |x| = n\}$. Then we can simply let the sparse set S, at length n, encode the circuit C_n in a canonical way. To show that A is decidable in polynomial time given the oracle S, a deterministic Turing machine upon input x, starts by extracting the circuit $C_{|x|}$, and then uses the circuit to decide whether $x \in A$.

To prove the other direction we extend the proof that the class P has small circuits. Thus suppose $A = L(M^S)$, where the running time of M is bounded by n^c, and S is sparse. In the proof for P, we lay down a grid of n^c by n^c, indexed by time step i and cell location j. Each grid node will be a finitary boolean gadget that determines, at time step i and cell location j, what the machine does there. The boolean gadget takes input a boolean variable indicating whether the read/write head is currently located at location j, and if so, the state q of the Turing machine and the current tape symbol a. The gadget produces the output simulating the Turing machine move at step i and location j. The output affects cell location $j-1$, j and $j+1$, at time $i+1$. (This is in fact a LOGSPACE reduction from P to the circuit value problem CVP [Lad75], which will be used later.)

Now the only modification for the machine M with oracle S is as follows: Let $S^{=n}$ be represented by a boolean formula s_n in disjunctive normal form. We append the boolean formulae s_ℓ, for $\ell \le n^c$ to the circuit described above, and augment each local gadget to take an additional input that indicates whether the machine enters its query state. An extra bit vector at each time step i can be used to store the queried string. And the query is answered by the formulae s_ℓ.

A related result of Pippenger [Pip79] is the following: The class of languages decidable in polynomial time with polynomial length advice is identical to the class of languages having polynomial-size circuits, thus also identical to the $\le_T^P$-closure of sparse sets.

2.2 Karp-Lipton theorem

We now come to the beautiful theorem of Karp and Lipton [KL82] with contributions by Sipser [Kar].

Theorem 2.2 *If* NP *has a sparse hard set S under polynomial time Turing reducibility $\leq_T^P$, then the polynomial time hierarchy collapses to its second level $\Sigma_2^p = \Pi_2^p$.*

Note that the condition in the theorem is equivalent to the assumption that every language in NP has polynomial size circuits.

By self-reducibility, given a boolean circuit for SAT, one can modify the circuit so that, upon input f, it will either output NO, meaning f is unsatisfiable, or output YES, with a satisfying assignment to f. This is accomplished by the most basic use of self-reducibility: If the circuit on f outputs YES, then it must answer YES to at least one of the sub-formulae $f_{x_1=0}$ and $f_{x_1=1}$. Take the one with a YES answer and repeat the process, till a satisfying assignment is obtained. We record this idea as a lemma for future use.

Lemma 2.1 *Any decision algorithm for* SAT*(or circuit for* $\mathrm{SAT}^{\leq n}$*) can be modified in polynomial time to find a satisfying assignment for any satisfiable formula (of size up to n).*

Now the proof of the Karp-Lipton theorem:

Suppose a circuit family of polynomial size for SAT exists. By the above observation, for all n, we assume a circuit of size n^c exists, such that for every f of size up to n, it will say NO when $f \notin$ SAT, or YES with a satisfying assignment to f, when $f \in$ SAT. Let L be an arbitrary language in Π_2^p. We will design a Σ_2^p-machine M to accept L, thus $\Pi_2^p \subseteq \Sigma_2^p$. The theorem follows from that.

By Cook's theorem [Coo71], there is a polynomial time function $\Gamma(\cdot,\cdot)$, such that,

$$L = \{x \mid (\forall^p y)\,[\Gamma(x,y) \in \mathrm{SAT}]\},$$

where $\forall^p y$ ranges over all y of length some polynomial in the length of x, say $|x|^d$. Note that all the boolean formulae $\Gamma(x,y)$ produced here are bounded by a polynomial in $|x|$, say $|x|^e$.

Upon input x, our Σ_2^p-machine M first guesses all boolean circuits of size $|x|^{ce}$. For each guess C, M treats it *as if* it will solve SAT up to length $|x|^e$ as described above. M then enters its universal stage, $\forall y$, $|y| = |x|^d$, it computes $\Gamma(x,y)$ and tries to use C to decide its satisfiability, and accepts on this path if and only if C says YES, and *it is verified* that the presumed satisfying assignment output by C indeed satisfies $\Gamma(x,y)$.

We claim that this Σ_2^p-machine M accepts L. First let $x \in L$. Since circuits for SAT of polynomial size exists, some guess will find such a circuit, and $\forall y$, $|y| = |x|^d$, it will say YES to $\Gamma(x,y)$, and the satisfying assignment will be verified. Hence M accepts x.

Next, assume M accepts x. Then for some guess C, *it is verified* that $\Gamma(x, y)$ are satisfiable for all y, $|y| = |x|^d$. (This is so, regardless whether a "correct" C was at hand.) Hence $x \in L$.

2.3 Long's extension

Suppose there is a boolean circuit of polynomial size for SAT, then with this circuit in hand, any NP problem can be solved "locally" for all instances of small size, as though it belongs to P. One can modify the same circuit to solve a Σ_2^p or Π_2^p problem "locally." Whether this is also true for sparse oracles is answered by the following theorem of Long [Lon82].

Theorem 2.3 *If there is a sparse set S, such that* NP $\leq_T^P S$, *then*

$$\mathrm{PH} = \Sigma_2^p = \Pi_2^p \leq_T^P S.$$

In other words, if NP has a sparse-hard set S under polynomial time Turing reducibility $\leq_T^P$, then the same sparse set S is hard for the entire polynomial time hierarchy.

Proof Let SAT be accepted by M^S. Let L be an arbitrary language in Π_2^p as before,

$$L = \{x \mid (\forall y, |y| = |x|^d)\,[\Gamma(x, y) \in \mathrm{SAT}]\}.$$

The idea is to try to get our hands on the sparse set S. If sparse set S is a tally set, namely $S \subseteq 1^*$, or otherwise P-printable, then it would have been easy. (This is the case, such as in the proof of Meyer's theorem, when we had a choice how to code the circuit in the sparse oracle.) In the general case, we may not be able to query all the right places in S, and obtain all the strings of S up to a polynomial length. Instead, we will be content with obtaining all the strings of S that "matter" in the local decision problem for the SAT instances $\Gamma(x, y)$.

Thus, we fix an x of length n. Initially set $\underline{S} = \emptyset$. With the current $\underline{S}$ in place of S, we can form a circuit C_n, using M just as in the proof of Meyer's theorem. This circuit will solve SAT for length n, *if* our current $\underline{S}$ agrees with S up to a length which is an appropriate polynomial of n. Also we make the circuit always check any purported satisfying assignment, thus the only error the circuit may make is to say NO to an $f \in \mathrm{SAT}$.

We formulate the following NP question:

> Is there a y, with $|y| = |x|^d$, such that $\Gamma(x, y) \in \mathrm{SAT}$, yet our circuit says NO?

Clearly this is an NP question, which we can solve deterministically running M^S. If the answer is NO, then we are done: the current $\underline{S}$ serves just as well as S *for all instances* $\Gamma(x, y)$ that matter to us. If the answer is YES, then we can use self reducibility as in Lemma 2.1 to extract an explicit y such that $\Gamma(x,y) \in$ SAT and yet our circuit says NO.

Now, with this y in hand we run M^S on $\Gamma(x,y)$ using the oracle S. Some queried strings by M must be in S but not in the current $\underline{S}$. Otherwise, the current circuit using $\underline{S}$ would have agreed with M^S on $\Gamma(x,y)$. Add those strings in $S - \underline{S}$ to $\underline{S}$. Now we can repeat the process using the updated $\underline{S}$.

This process of updating $\underline{S}$ to approximate S can only go on at most a polynomial number of steps, since $\underline{S}$ always remains a subset of S, and each time we add some string of polynomial length from S to $\underline{S}$.

Once $\underline{S}$ gives us a circuit correct for all instances of $\Gamma(x,y)$, we can treat the questions of whether "$[\Gamma(x,y) \in \text{SAT}]$?" as a polynomial time computation, and thus $(\forall y, |y| = |x|^d)[\Gamma(x,y) \in \text{SAT}]$ becomes a coNP question that we can compute using M^S. □

Long's theorem has the following corollary.

Corollary 2.1 *If there exists a sparse set* $S \in \Delta_2^p$ *such that* $\text{NP} \leq_T^P S$, *then* $\text{PH} = \Delta_2^p$.

Very recently, Bshouty et al. [BCG+95] (see also [KW95]) show that polynomial size circuits are learnable in ZPP with equivalence queries and with additional queries to NP, from which it follows that polynomial size circuits for SAT is learnable in ZPP^{NP}, and thus, sparse $\leq_T^P$-hard sets for NP collapse the polynomial time hierarchy to ZPP^{NP}.

Theorem 2.4 *If sparse* $\leq_T^P$*-hard sets exist for* NP, *then* $\text{PH} = \text{ZPP}^{\text{NP}}$.

We also mention that for $\leq_T^P$-completeness, Kadin [Kad89] show that PH collapses to the class $\text{P}^{\text{NP}[\log]}$, the class of languages decidable in polynomial time with $O(\log n)$ queries to an NP oracle.

Theorem 2.5 *If sparse* $\leq_T^P$*-complete sets exist for* NP, *then* $\text{PH} = \text{P}^{\text{NP}[\log]}$.

Improving the Karp-Lipton collapse to P = NP, or even NP = coNP appears hard. We note that the optimality of these results in terms of oracles have been proven [Hel86, IM89, Kad89].

3 Earlier results for many-one reductions

3.1 The isomorphism conjecture for NP

As we mentioned earlier, one of the major motivations for the study of sparse sets is the isomorphism conjecture of Berman and Hartmanis [BH77]. It says that all NP-complete languages under $\leq_m^P$ are isomorphic. The conjecture will imply, among other things, $\text{NP} \neq \text{P}$.

Berman and Hartmanis showed in 1976, that all known NP-complete languages at the time under polynomial time many-one reduction $\leq_m^P$ are isomorphic. This means that there is a polynomial time computable function σ from Σ^* to Σ^*, that is one-to-one, onto, and its inverse is also polynomial time computable, such that σ maps one NP-complete language to another. Their proof relies on the notion of a padding function, and is an adaptation of the classical Cantor-Bernstein theorem in set theory: if there are one-to-one functions from set A to set B, and vice versa, then there is a one-to-one correspondence between A and B.

It was observed in [BH77] that the usual NP-complete languages L, such as those found in [GJ79], all admit a padding function:

$$\pi : \quad \Sigma^* \times \Sigma^* \longrightarrow \Sigma^*$$

satisfying the following conditions: 1) $|x|+|y| < |\pi(x,y)|$, 2) given x and y, $\pi(x,y)$ is polynomial time computable, 3) given $\pi(x,y)$, it is also polynomial time computable to extract x and y uniquely, (this implies that π is one-to-one), and finally 4) for all x and y, $x \in L \Leftrightarrow \pi(x,y) \in A$. For instance, in the case of SAT, one can append a score of dummy variables and clauses to a boolean formula and not change its satisfiability.

The use of such a padding function is to make a reduction length increasing. Thus, for a given reduction $\sigma : A \leq_m^P B$, we can modify it to be $x \longmapsto \pi_B(\sigma(x), 1^{|x|})$, provided a padding function π_B exists for B.

Suppose now there are two length increasing reductions $\sigma : A \leq_m^P B$ and $\tau : B \leq_m^P A$. Berman and Hartmanis [BH77] show how to obtain a polynomial time isomorphism between A and B.

Denote by Σ and Γ respectively the alphabets over which the languages A and B are defined (Σ and Γ could be identical of course). Define $\Sigma_0 = \Sigma^*$ and $\Gamma_0 = \Gamma^*$. Define $\Sigma_i = \tau(\Gamma_{i-1})$ and $\Gamma_i = \sigma(\Sigma_{i-1})$ for $i \geq 1$. Clearly $\Sigma_0 \supset \Sigma_1 \supset \Sigma_2 \supset \cdots$ and similarly $\Gamma_0 \supset \Gamma_1 \supset \Gamma_2 \supset \cdots$. Since the reductions are one-to-one and length increasing, the containments are strict. Also by the length increasing property, $\bigcap_{i=0}^{\infty} \Sigma_i = \emptyset$ and

$$\Sigma^* = \bigcup_{i=0}^{\infty} [\Sigma_i - \Sigma_{i+1}].$$

A similar decomposition holds for Γ^*.

Given any $z \in \Sigma^*$, let i_z be the largest $i \geq 0$ such that $z \in \Sigma_i$, i.e., $z \in \Sigma_{i_z} - \Sigma_{i_z+1}$. Because the reductions are length increasing, and polynomial time invertible, we can decide i_z in polynomial time. The same is true for any string in Γ^*.

Now define a map $\rho : \Sigma^* \longrightarrow \Gamma^*$. Given $z \in \Sigma^*$, if i_z is even, then define $\rho(z) = \sigma(z)$; if i_z is odd, then let $\rho(z) = \tau^{-1}(z)$. Thus, we set up a one-to-one correspondence in which the following terms are term by term mapped to each other:

$$\Sigma^* = [\Sigma_0 - \Sigma_1] \cup [\Sigma_1 - \Sigma_2] \cup [\Sigma_2 - \Sigma_3] \cup [\Sigma_3 - \Sigma_4] \cup \ldots,$$

and

$$\Gamma^* = [\Gamma_1 - \Gamma_2] \cup [\Gamma_0 - \Gamma_1] \cup [\Gamma_3 - \Gamma_4] \cup [\Gamma_2 - \Gamma_3] \cup \ldots.$$

It can be easily verified that the map ρ is an isomorphism from A to B.

From a structural complexity point of view, the theorem of Berman and Hartmanis and their isomorphism conjecture, if true, are esthetically pleasing. If we consider polynomial time isomorphisms as merely renaming of inputs within polynomial time, then the theorem says that all these NP-complete problems in [GJ79], while varied and fascinating combinatorially, are essentially one and the same problem.

However, there is now considerable doubt whether the Berman-Hartmanis isomorphism conjecture is true [JY85]. The isomorphism conjecture is an active research topic with considerable work on the subject. (For the current status of the subject, see [FFK92, KMR90, You90].) We will not go into this any further in this article.

3.2 Mahaney's theorem

As typical NP-complete languages such as SAT are all exponentially dense, and polynomial time isomorphisms cannot change the density from exponential to polynomial, it follows from the isomorphism conjecture that no sparse set can be NP-complete under $\leq_m^P$. This becomes the sparse non-completeness conjecture for NP by Berman and Hartmanis. It was resolved by the following definitive theorem of Mahaney [Mah82].

Theorem 3.1 *There are no sparse* NP*-complete (or* NP*-hard) languages under* $\leq_m^P$ *if and only if* NP $\neq$ P.

Mahaney's theorem was the culmination of considerable earlier efforts toward the sparse non-completeness conjecture of Berman and Hartmanis. In particular we mention that P. Berman [Ber78] was the first to obtain a partial result in this direction. He showed that NP-hard sets over one-letter alphabet exist if and only if P = NP. This was subsumed by a result of Fortune [For79], who showed that sparse coNP-hard sets collapse NP to P. Note that if $S \subseteq 1^*$ is NP-hard, then the complement $1^* - S$ of S within 1^* becomes a coNP-hard sparse set.

Theorem 3.2 *Sparse* coNP*-hard sets exist if and only if* P = NP.

While the Karp-Lipton theorem handles the case of polynomial time Turing reductions, and Mahaney's theorem settles for the many-one case, there have been numerous attempts to try to bridge the gap between the two reductions. In particular, quite a few partial results toward the case of bounded truth-table reduction were obtained [Yap83, Ukk83, Wat92, Yes83]. In 1990, Ogihara and Watanabe [OW91] settled this case completely. We will discuss their proof in the next section in detail; here we will give a simultaneous and simple proof of both Mahaney's theorem and Fortune's theorem, using a tool introduced by Ogihara and Watanabe, called left-sets.

Let L be a language in NP, let p be a polynomial and A be a set in P such that

$$x \in L \iff (\exists w : |w| = p(|x|))\,[\langle x, w\rangle \in A]. \tag{1.1}$$

Define the *left-set* with respect to p and A, denoted by $Left[p, A]$ as the set of all $\langle x, y\rangle$, $|y| \le p(|x|)$, such that

$$(\exists w : |w| = p(|x|), w \ge y)\,[\langle x, w\rangle \in A]. \tag{1.2}$$

Here y is a node and w is a leaf in the full binary tree of depth $p(|x|)$, and $>$ denotes the dictionary order; that is, a node u is smaller than v if and only if either v is a descendant of u or lies to the right of u. Thus, $\langle x, y\rangle \in Left[p, A]$ if and only if the right-most leaf w, $|w| = p(|x|)$, such that $\langle x, w\rangle \in A$, either lies to the right of y or through the node y.

Clearly, $Left[p, A]$ is in NP and $L \le_m^P Left[p, A]$, by the map $x \mapsto \langle x, \epsilon\rangle$. The set $Left[p, A]$ embodies a certain "totally ordered self-reducible structure:" For every x and every pair $y, y' \in \Sigma^{\le p(|x|)}$ with $y' > y$, it holds that:

$$\langle x, y'\rangle \in Left[p, A] \Rightarrow \langle x, y\rangle \in Left[p, A]. \tag{1.3}$$

For readers who are familiar with the notion of a P-selective set, there is more than a passing resemblance here. A set S is called P-selective, if there is a selector s, which is polynomial time computable, such that for any strings x and y, $s(x, y) \in \{x, y\}$, and if $\{x, y\} \cap S \ne \emptyset$ then $s(x, y) \in S$. In other words, the selector selects the "more likely" candidate among the two input strings for membership in S. Clearly for a P-selective set S there is a (quasi-) ordering for membership in S, such that one can compare any two given strings in this ordering in polynomial time. For any NP-complete language, the left-set $Left[p, A]$ can be viewed as introducing a certain "local selectivity" (1.3) while remaining NP-complete. [1]

Now we prove the theorems of Mahaney and Fortune using left-sets. First, for Mahaney's result, let us assume NP $\le_m^P S$, a sparse set. Then for SAT, its left-set $Left[p, A]$ is $\le_m^P$-reducible to S via some reduction f,

[1] The genesis of the idea of a left set can be found in a paper by G. Miller [Mil76] where he defined the following set for a partial function f

$$R_f = \{(x, y) \mid x \in dom(f) \text{ and } y \le f(x)\}.$$

Similar notions were also used in the study on public-key cryptosystems by Grollmann and Selman [GS88].

where A consists of pairs $\langle \phi, \sigma \rangle$ such that ϕ is a boolean formula, and σ is a satisfying assignment for ϕ. For any ϕ and a partial assignment τ that assigns boolean values to the first k variables for some $k \geq 0$, we assume the image of the reduction $f(\langle \phi, \tau \rangle)$ is bounded in length by a polynomial $p(|\phi|)$. Since S is sparse, we can assume there are at most $q(|\phi|)$ many strings in S among $f(\langle \phi, \tau \rangle)$, where $q(\cdot)$ is another polynomial.

Now we will visit the partial truth assignments τ to ϕ in a breadth-first-search starting with the root, where throughtout the search, the nodes we maintain are all on the same level. When we reach a level with more than $q(|\phi|)$ many nodes, we start pruning. We compute for each τ on this level, a label $f(\langle \phi, \tau \rangle)$. Whenever there are two labels $f(\langle \phi, \tau \rangle)$ and $f(\langle \phi, \tau' \rangle)$ that are identical, we conclude that $\langle \phi, \tau \rangle \in \mathit{Left}[p, A]$ if and only if $\langle \phi, \tau' \rangle \in \mathit{Left}[p, A]$. Suppose τ is to the left of τ', $\tau < \tau'$. We observe that *the right-most* satisfying assignment, if one exists, must not pass through τ. Thus, we can safely eliminate the node τ and its subtree from further consideration. After the elimination of all duplicate labeled nodes, if we end up with no more than $q(|\phi|)$ many nodes at that level, we go on to the next level. If, on the other hand, we still have more than $q(|\phi|)$ many distinct labels $f(\langle \phi, \tau \rangle)$, we conclude that one of them must be outside of S. Thus one of the pairs $\langle \phi, \tau \rangle$ must be outside of $\mathit{Left}[p, A]$. However, since we have "local selectivity" (1.3), the right most τ among these must be outside of $\mathit{Left}[p, A]$, thus it can be safely eliminated. In any case, we end up with no more than $q(|\phi|)$ many nodes at this level. If we have not reached the leaves, we extend each remaining node by its two children, to arrive at the next level.

Inductively the pruning process from one level to the next satisfies the following property: If at any level, the right most satisfying assignment exists and passes through one of the labeled nodes, then the same is true for the next level of labeled nodes after the pruning.

When the partial assignments are fully extended, we reach the leaves of the assignment tree. At the leaves we can directly check whether any of the assignments satisfies the boolean formula ϕ. Note that followed from the induction, $\phi \in \mathrm{SAT}$ if and only if the right most satisfying assignment exists and *is* one of the leaves. This gives a polynomial time algorithm for SAT, assuming the existence of an NP-complete (or NP-hard) sparse set. This proves Mahaney's theorem 3.1.

A similar argument works for Fortune's theorem. The only change one makes is that when the number of distinct labels exceeds $q(|\phi|)$, we already know one of the strings is in S^c, and thus one of the pair $\langle \phi, \tau \rangle \in \mathit{Left}[p, A]$. Hence $\phi \in \mathrm{SAT}$.

4 Bounded truth table reduction of NP

We have seen that the left-sets introduced in [OW91] give new and uniform proofs of earlier results. Not only that, using the new technique Ogihara and Watanabe succeeded in proving the following theorem that unifies many previous results:

Theorem 4.1 *If* NP $\leq_{btt}^{P}$ S*, a sparse set, then* NP = P*.*

Suppose there is a sparse $\leq_{btt}^{P}$-hard set for NP. The left-set $L(\mathrm{SAT})$ of SAT is $\leq_{k\text{-}tt}^{P}$-reducible to a sparse S via some function f for some k. This means for every boolean formula ϕ and a partial assignment τ, $f(\langle\phi,\tau\rangle)$ produces a list of strings $y_1, y_2, \ldots, y_k$, and a truth table α on k inputs, such that $\langle\phi,\tau\rangle \in L(\mathrm{SAT})$ if and only if the characteristic sequence of membership of y_i in S satisfies α:

$$\alpha(\chi_S(y_1), \chi_S(y_2), \ldots, \chi_S(y_k)) = 1.$$

There are 2^{2^k} many truth tables on k inputs. We will first consider a fixed truth table α. The theorems of Mahaney and Fortune are two special cases for 1-truth table reductions. Furthermore, in both cases, a fixed 1-truth table was used: the identity function for Mahaney, and the negation function for Fortune. In general, however, a truth table reduction produces a truth table that is also dependent on the input.

We will present a proof based on an idea of Homer and Longpré [HL94]. We will try to inductively build a pruning strategy that prunes any given set T of nodes at a fixed level of the assignment tree, given a fixed k-truth table α. The k-tt strategy will utilize the $(k-1)$-tt strategy. Each node in T produces a k-tuple $\langle y_1, y_2, \ldots, y_k\rangle$ by f. We collect those k tuples with an identical ith entry y_i, say y_{i0}, for any fixed i. Note that, given the membership $\chi_S(y_{i0})$, α is effectively reduced to a $(k-1)$-truth table on this subset. Of course, we don't know $\chi_S(y_{i0})$. However, if our inductively built $(k-1)$-tt pruning strategy is *oblivious*, i.e., is independent of the particular fixed $(k-1)$-truth table used, then we would be able to carry through.

Let us first consider 1-truth tables. Suppose there are at most $q(n)$ many strings of S that can possibly appear among the y's, due to length and sparsity. The pruning strategies we presented for the Mahaney and Fortune theorems are not quite oblivious. The non-oblivious case is when the number of distinct labels after the elimination of duplicate ones still exceeds $q(n)$. However, suppose we modify it as follows. We will first visit all the given nodes T at this level from left to right, eliminating duplicate labeled nodes as before. If there are more than $q(n)$ nodes left, $v_0 < v_1 < \ldots < v_{q(n)}$, we stop and then start to visit T from right to left. During this right-to-left sweep we also eliminate duplicate labeled nodes in the same way. If we encounter $v_{q(n)}$ before seeing more than $q(n)$ distinct labels, we stop, and pass to the next level with at most $2q(n)+1$ nodes. This becomes our new bound on the width of the pruning procedure. Suppose now the right-to-left sweep finds more than $q(n)$ distinctly labeled nodes $u_{q(n)} < \ldots < u_1 < u_0$, before encountering $v_{q(n)}$. In this case, $v_{q(n)}$ still lies to the left of $u_{q(n)}$. The key observation is the following: For some i and j, where $0 \leq i, j \leq q(n)$, the reduction f must map both v_i and u_j out of S. Thus the 1-truth table takes the same value at v_i and u_j, *regardless* of what truth table is being used, be it the identity function or the negation function, or whatever. This is the obliviousness we are looking for. Thus,

$$\langle \phi, v_i \rangle \in L(\mathrm{SAT}) \Longleftrightarrow \langle \phi, u_j \rangle \in L(\mathrm{SAT}).$$

It follows that, *it cannot be the case that the right most satisfying assignment passes through a node w, such that $v_i \leq w < u_j$.* Thus all nodes w in T such that $v_i \leq w < u_j$ can be eliminated. Of course v_i and u_j are unknown to us. But we can safely eliminate all $w \in T$ such that $v_{q(n)} \leq w < u_{q(n)}$ because $v_i \leq v_{q(n)}$ and $u_{q(n)} \leq u_j$. So, we end up with at most $2q(n) + 1$ nodes again. Inductively from one level to the next in the assignment tree, we maintained that

> (*) If at any level, the right most satisfying assignment exists and passes through one of the given nodes, then the same is true after the pruning.

This completes the proof of the case for any fixed 1-truth table, and the pruning strategy is now oblivious of the particular truth table used.

We are now ready to tackle the case with a fixed k-truth table α. We would like to arrive at an *oblivious* pruning strategy $\mathcal{S}_k$ that satisfies the above inductive requirement (*), and leaves at most a polynomial number of nodes $p_k(n)$ after the pruning. We already see that we can set $p_1(n) = 2q(n)+1$. In general, suppose an oblivious pruning strategy $\mathcal{S}_{k-1}$ has been found that works for any fixed $(k-1)$-truth table.

Given any set of nodes T on any level. Let each node w in T be labeled by a k-tuple $\langle y_1(w), y_2(w), \ldots, y_k(w)\rangle$ via f. For any $w \in T$, any i, $1 \leq i \leq k$, consider the subset

$$T_{i,w} = \{u \in T \mid y_i(u) = y_i(w)\}.$$

Note that if $y_i(w) = y_i(w')$, then $T_{i,w} = T_{i,w'}$, i.e., the set $T_{i,w}$ depends only on i and $y_i(w)$, and not on w. Also note that depending on $\chi_S(y_i(w)) = 0$ or 1, the truth table α induces one of two possible fixed $(k-1)$-truth tables on the subset $T_{i,w}$. Employing the pruning strategy $\mathcal{S}_{k-1}$ we can prune this subset down to size at most $p_{k-1}(n)$.

What if the total number of remaining nodes after pruning for all w and i is still too large? We will show that, in this case, *some* node v remaining must have all its $y_i(v) \notin S$, $1 \leq i \leq k$.

Indeed, suppose for every v there exists an i, such that $y_i(v) \in S$. Then we claim there could be at most $kq(n)p_{k-1}(n)$ nodes left. This can be seen as follows. Let $\widetilde{T}_{i,w}$ be the set of nodes remaining from $T_{i,w}$ after applying $\mathcal{S}_{k-1}$, and let $\widetilde{T}$ be the set of nodes remaining from T after pruning for all i and all w. If v still remains, then $v \in \widetilde{T}_{i,v}$, where i is an index such that $y_i(v) \in S$. Hence

$$\widetilde{T} \subseteq \bigcup_{y \in S} \bigcup_{i: y_i(v)=y} \widetilde{T}_{i,v}.$$

We have used the fact that the set $T_{i,v}$ only depends on i and $y_i(v)$. Since each $|\widetilde{T}_{i,v}| \le p_{k-1}(n)$, there can be at most $kq(n)p_{k-1}(n)$ nodes left.

Hence, if $|\widetilde{T}| > kq(n)p_{k-1}(n)$, then for some $w \in \widetilde{T}$, $y_i(w) \notin S$, for all $1 \le i \le k$.

Let $p_k(n) = 2kq(n)p_{k-1}(n) + 1$. It is easily verified that

$$p_k(n) \le k! \cdot (2q(n)+1)^k.$$

At this point we are at a similar position as in the 1-truth table case after the elimination of all duplicate labels. We will carry out two sweeps: first a left-to-right sweep, and if more than $kq(n)p_{k-1}(n)$ many nodes remain, then a second right-to-left sweep. If we end up with at most $p_k(n)$ many nodes total after the two sweeps, we are done. Otherwise those nodes between the $(kq(n)p_{k-1}(n)+1)$st nodes from the left, and the $(kq(n)p_{k-1}(n)+1)$st nodes from the right can be safely eliminated. (The interval is inclusive on the left and exclusive on the right.) This completes the description of the strategy $\mathcal{S}_k$.

Finally to handle the situation where the truth tables are not fixed, we simply note that there can be at most 2^{2^k} distinct k-truth tables. Thus we can carry out the above pruning strategy 2^{2^k} times in parallel for each fixed k-truth table on any given level of the assignment tree. The theorem of Ogihara and Watanabe is proved.

4.1 Extensions

The breakthrough by Ogihara and Watanabe has provoked a flurry of results about sparse hard/complete set problems [HL94, AHH$^+$93, AKM92b, AKM92a, AA95, RR92, OL93] (see [HOW92] for a survey). Below we state the best known results on polynomial time sparse hard sets for NP.

Theorem 4.2 *1. [AKM92b]* NP *is included in the $\le^P_{btt}$-reducibility closure of the languages that are $\le^P_{ctt}$-reducible to sparse sets if and only if* P = NP.

2. [AKM92a] Sparse $\le^P_{dtt}$-hard sets for NP *exist only if* $\mathrm{PH} = \Delta^p_2$.

3. [HL94] Sparse $\le^P_{O(\log n)\text{-}tt}$-hard sets for NP *exist only if* NP *is included in* $\mathrm{DTIME}[2^{O(\log^2 n)}]$.

Note that Part 1 in the above subsumes the Ogihara-Watanabe theorem. We note here that very recently, this part has been slightly strengthend by Agrawal and Arvind [AA95]: they showed that NP $\subseteq \mathcal{C} \Rightarrow$ P = NP holds for a circuit class $\mathcal{C}$ provably larger than the "$\leq_{btt}^{P}$ of $\leq_{ctt}^{P}$" closure of sparse sets in Part 1.

Does the left-set technique shed light on the sparse hard/complete set problems for other counting classes? It is easy to see that the "breadth-first-search plus tree-pruning" method for the largest satisfying assignment works for UP and FewP, subclasses of NP. Also, it is easy to see that a statement similar to Theorem 2.2 holds for every complexity class having complete sets with some reasonable self-reducible structure, and thus, if such a complexity class $\mathcal{C}$ includes either NP or coNP, then sparse $\leq_{btt}^{P}$-hard sets for $\mathcal{C}$ collapses $\mathcal{C}$ to P. Generally speaking, for every counting class in the sense of [GNW90], sparse $\leq_{btt}^{P}$-hard sets for the class collapse it within NP $\bigcap$ coNP [OL93]. The story for modulo-based counting complexity classes such as $\oplus$P is slightly different, for it is not known whether the class contain NP or coNP. Ogihara and Lozano [OL93] extend the notion of left-sets and show that sparse $\leq_{btt}^{P}$-hard sets for a modulo-based counting class collapse it to P. This framework allows one to obtain a clone of Theorem 4.2 (1) for many counting classes.

Theorem 4.3 *For each* $\mathcal{C}$ *chosen from*

$$\mathrm{PSPACE}, \mathrm{UP}, \mathrm{FewP}, \mathrm{C_{=}P}, \mathrm{PP}, \mathrm{Mod_2P}, \mathrm{Mod_3P}, \cdots,$$

$\mathcal{C}$ *is included in the* $\leq_{btt}^{P}$*-reducibility closure of the languages that are* $\leq_{ctt}^{P}$*-reducible to sparse sets if and only if* $\mathrm{P} = \mathcal{C}$.

5 The Hartmanis conjecture for P

We now come to the Hartmanis conjecture on the non-existence of sparse complete sets for P under logspace computable many-one reductions. This conjecture, unlike its NP analog, remained open for many years. In a breakthrough Ogihara showed that the existence of P-hard sparse sets implies that P $\subseteq$ DSPACE[$\log^2 n$] [Ogi95]. Ogihara defined an ingenious auxiliary language, which is P-complete and has a certain built-in self-reducibility. The proof then proceeds by simulation of P in DSPACE[$\log^2 n$] using the P-hard sparse set.

This was followed by the decisive improvement of Cai and Sivakumar [CS95a] that resolved the conjecture for P. The Cai-Sivakumar proof uses algebra extensively. Parallel computation and randomization play a crucial role as well. The starting point is the language defined by Ogihara. This language enables one to set up linear equations to solve for CVP, the circuit value problem. The backbone of the proof is the formation and solution of certain linear equation systems over finite fields.

It turns out to be natural to employ randomization here to form the equations. Then, it is possible to prove that the system has a sufficiently high rank, with high probability. With a lower bound on the rank, one can then appeal to the NC^2 algorithm of Mulmuley [Mul87] to solve the linear system. This gives a randomized NC^2 algorithm for CVP. Thus, both randomization and parallel complexity class NC enters the picture quite naturally, even though no mention of either subjects is made in the original conjecture.

The next step in the Cai-Sivakumar proof is derandomization which implies that $\mathrm{P} = \mathrm{NC}^2$, given a P-hard sparse set exists. Here the main tools are finite fields and polynomial size sample spaces. It is shown that a deterministic construction D exists such that if we exhaustively sample from D, it is as good as if we are taking uniform samples from an exponentially larger sample space. It follows that the probabilistic conclusions are now always true, thus getting rid of randomness in the proof. This result also provides an alternative proof of Ogihara's theorem, since $\mathrm{NC}^2 \subseteq \mathrm{DSPACE}[\log^2 n]$. Note that since the proof goes through the parallel class NC^2, proving the inclusion in $\mathrm{DSPACE}[\log^2 n]$ does not involve any step-by-step automata theoretic simulation of CVP.

Finally, Cai and Sivakumar improved their proof further from NC^2 down to NC^1 to resolve Hartmanis's conjecture. We will start with the language constructed by Ogihara and the randomized NC^2 algorithm. In the following, all NC classes are logspace uniform, meaning that given 1^m, a logspace computation will produce the mth circuit for input length m.

5.1 Ogihara's language and randomized NC^2

To begin with, suppose a sparse set S exists that is hard for P under logspace many-one reductions. Recall that a typical P-complete language is the circuit value problem CVP. In [Ogi95], Ogihara defined a set A closely related to CVP: It consists of tuples $\langle C, x, I, b\rangle$, where C is a boolean circuit, x is an input to C, I is a subset of the gates, and b is a bit (0 or 1), such that the sum mod 2 of the values of the gates chosen in I from C on input x equals b, i.e.,

$$\bigoplus_{i \in I} g_i(x) = b.$$

Clearly, $A \in \mathrm{P}$ and hence $A \leq_m^L S$. Let f be a logspace computable function such that for all x, $x \in A \iff f(x) \in S$. It is also obvious that $\mathrm{CVP} \leq_m^L A$.

Our goal here is to show that A can be solved in randomized NC^2. Here NC^2 denotes the class of languages computed by uniform family of circuits of polynomial size and $O(\log^2 n)$ depth.

We note that for any C, x, I, exactly one of the bits $b = 0, 1$ satisfies the equation, and thus exactly one of $f(\langle C, x, I, 0\rangle)$ and $f(\langle C, x, I, 1\rangle)$ is a string in S. Moreover, suppose for two *distinct* subsets I and J and some pair of bits b, b', $f(\langle C, x, I, b\rangle) = f(\langle C, x, J, b'\rangle)$, (we are not assuming that the image is in S). In this case, regardless of whether $\bigoplus_{i \in I} g_i(x) = b$ and $\bigoplus_{i \in J} g_i(x) = b'$ are true or not, they hold or fail simultaneously. Thus we have an equation mod 2 on the values of the gates of C on input x, namely

$$\bigoplus_{i \in I \triangle J} g_i(x) = b \oplus b', \tag{1.4}$$

and $I \triangle J \neq \emptyset$.

Fix any C and x, let n denote the number of nodes in C (including the inputs, output, and the interior gates). Let $p(n)$ be a polynomial bound on the number of strings in S among $f(\langle C, x, I, b\rangle)$, (over all I and b). Thus, some string $w \in S$ must be mapped on by at least $2^n/p(n)$ many subsets I, more precisely, by the tuple $\langle C, x, I, b_I\rangle$, where b_I is the "right value" $b_I = \bigoplus_{i \in I} g_i(x)$. As described above, any two such I give rise to an equation mod 2 on the values of the gates of C on input x. The idea now is to choose polynomially many random subsets $I \in \{0,1\}^n$ and compute $f(\langle C, x, I, 0\rangle)$ and $f(\langle C, x, I, 1\rangle)$, collecting as many equations as possible. The following lemma ensures that this process gives us a system of linear equations of sufficiently high rank, even if we restrict attention to a single "popular" $w \in S$ which appears for at least $2^n/p(n)$ many subsets I.

A probabilistic lemma

Let $B = \{0,1\}^n$ denote the n-dimensional binary cube. With respect to the finite field of two elements $GF(2) = \mathbf{Z}_2$, B is a vector space of dimension n. We note that when we obtain an equation of the form (1.4), it is on the symmetric difference of two subsets I and J, which are found to have the same reduction value. In terms of the vector space, the coefficient vector of the equation (1.4) is the difference of the two boolean vectors in B that represent I and J. Intuitively, fix any "popular" $w \in S$ which appears for many subsets I, we can expect a good many randomly sampled points to be mapped to w, which in turn give us equations with the coefficient vectors being the differences of such pairs.

Let $T \subseteq B$ be an arbitrary subset of the cube. We ask the following question: If we uniformly and independently pick a sequence of m points in B, what can we say about the probability distribution of the dimension of the affine span of those points picked from T as a function of m, n, and $|T|$? Here the affine span of a set T' is the smallest dimensional affine subspace that contains T'. It is a translation of a linear subspace which is generated by all the differences of pairs of vectors from T'. The dimension of the affine span of the set of points picked by the random process which have the same reduction value w is a lower bound on the rank of our linear equation system.

Lemma 5.1 *Suppose $|T| \geq 2^n/k$, where $k = n^{O(1)}$, then for $m = 2kn^2 + n = n^{O(1)}$, if we uniformly and independently pick a sequence of m points* *in B, the probability that the dimension of the affine span of the points from T is less than $n - \log_2 k$ is at most $e^{-n^2+O(n \log n)}$.*

We omit the proof here. The interested reader can find it in [CS95a].

The randomized NC^2 algorithm

By the above lemma, if in parallel we try polynomially many uniformly and independently chosen I, with high probability we will obtain a system of linear equations with rank deficiency at most $\log_2 p(n)$. We now describe how we can use these to determine in NC^2 the outputs of all the gates of C on input x.

Without loss of generality, let the rank of the system be $n - \log_2 p(n)$, and let $m (= n^{O(1)})$ denote the number of equations we have. Denote the equations by $E_1, \ldots, E_m$, and for $i \geq 1$, call an equation E_i *useful* if the rank $rk(E_1, \ldots, E_i) > rk(E_1, \ldots, E_{i-1})$. Clearly the number of useful equations is $n - \log_2 p(n)$. Mulmuley [Mul87] gives an algorithm to compute the rank of an $\ell \times n$ matrix, which, for $\ell = n^{O(1)}$, can be implemented by a circuit of depth $O(\log^2 n)$ and size $n^{O(1)}$. For $1 \leq i \leq m$, we compute in parallel $rk(E_1, \ldots, E_i)$, and identify all the useful equations. Now we have $n - \log_2 p(n)$ equations in n variables, with rank $n - \log_2 p(n)$. We apply the same process to the columns, and identify the $(n - \log_2 p(n))$-many useful columns. We rename the variables so that the first $n - \log_2 p(n)$ columns are all useful. For each of the $p(n)$ possible assignments to the last $\log_2 p(n)$ variables, we create in parallel a system of $n - \log_2 p(n)$ equations as an $(n - \log_2 p(n)) \times (n - \log_2 p(n))$ matrix. Each one of these can be solved in $\log^2 n$ depth and $\mathrm{poly}(n)$ size using the algorithm due to Borodin, et al. [BvzGH82]. For each potential solution we get for the gates of the circuit C on input x, we can check its validity using the local information about the circuit C and input x, such as $x_i = 0$, or $x_i = 1$, or $g_j(x) = g_k(x) \wedge g_\ell(x)$, etc. There will be a unique solution that passes all such tests and we will find the output of $C(x)$ in particular. We have proved:

Theorem 5.1 *If there is a sparse set that is hard for* P *under logspace many-one reductions, then* $\mathrm{P} \subseteq$ *Randomized* NC^2.

5.2 Deterministic construction

As before we have $B = \{0,1\}^n = \mathbf{Z}_2^n$ considered as an n-dimensional vector space over the finite field $\mathbf{Z}_2$. For each $I \in B$, let $b_I = \bigoplus_{i \in I} g_i(x)$ be the "right value." Then the string $w = f(\langle C, x, I, b_I \rangle) \in S$ and this w is called the color of I. The presumed reduction to the sparse set S gives a coloring of B with at most $p(n)$ colors. Let $D \subseteq B$ be a subset of B of cardinality bounded by a certain polynomial in n. The coloring of B induces a coloring of D, thus D is the union of at most $p(n)$ many color classes:

$$D = C_1 \cup C_2 \cup \ldots \cup C_{p(n)}.$$

Let the affine span of C_i be denoted by $L_i + d_i$, where L_i is a linear subspace, and d_i is a displacement vector. Let $L = L_1 + L_2 + \ldots + L_{p(n)}$ be the sum of the linear subspaces. We call L the span of the color classes. L_i is spanned by differences of vectors in C_i. For some spanning set of vectors of L_i, each vector in the set gives us an equation mod 2 of the values of the gates of C with the given input. If we collect a generating set of vectors for each L_i, together they span L. Thus, if we can construct a set D with polynomial size and with $\dim L \geq n - O(\log n)$ (irrespective of the coloring), we would have succeeded in derandomizing the construction of the last section. That is, by sampling exhaustively in D, we would have obtained a system of linear equations of rank $\geq n - O(\log n)$.

We claim that the above task can be accomplished as follows: given $p(n)$, construct a set D of polynomial size such that for any linear subspace M of B with $\dim M < n - \log_2 p(n)$, and any $p(n)$ displacement vectors $b_1, \ldots, b_{p(n)} \in B$, the union of the $p(n)$ affine subspaces $\bigcup_{i=1}^{p(n)} (M + b_i)$ does not cover the set D. For if so, then no matter what the induced coloring on D is, the span of the color classes L must be of dimension $\geq n - \log_2 p(n)$, simply because the union of at most $p(n)$ affine subspaces $\bigcup_{i=1}^{p(n)} (L + d_i)$ does cover D:

$$\bigcup_{i=1}^{p(n)} (L + d_i) \supseteq \bigcup_{i=1}^{p(n)} (L_i + d_i) \supseteq D.$$

Let $k = 1 + \log_2 p(n) = O(\log n)$. Without loss of generality, we may assume such a linear subspace M has dimension exactly $= n - k$. Any such M can be specified as the null space of a linear system of equations

$$a_{i1}x_1 + a_{i2}x_2 + \ldots + a_{in}x_n = 0,$$

where $i = 1, \ldots, k$, and the k vectors $\{(a_{i1}, a_{i2}, \ldots, a_{in}) \mid i = 1, \ldots, k\}$ are independent vectors in B over $\mathbf{Z}_2$.

Let $m = 2k + \log_2 n + 1 = 2\log_2 p(n) + \log_2 n + 3 = O(\log n)$. The Galois field $\mathbf{F} = GF(2^m)$ has a vector space structure over $GF(2)$ of dimension m. Choose any basis $\{e_1, \ldots, e_m\}$, then for $u = \sum_{i=1}^m u_i e_i$ and $v = \sum_{i=1}^m v_i e_i$ in $\mathbf{F}$, we can define an inner product by letting

$$\langle u, v \rangle = \sum_{i=1}^{m} u_i v_i,$$

and doing all arithmetic over $\mathbf{Z}_2$.

The (multi-) set D is defined as follows:

$$D = \{(\langle 1, v \rangle, \langle u, v \rangle, \ldots, \langle u^{n-1}, v \rangle) \mid u, v \in \mathbf{F}\}.$$

Note that $|D| = 2^{2m} = n^{O(1)}$. Now consider any non-zero vector $a = (a_0, a_1, \ldots, a_{n-1}) \in B$ and any $b \in \mathbf{Z}_2$. We claim that the affine hyperplane

$\sum_{i=0}^{n-1} a_i x_i = b$ cuts D into roughly two equal parts. To show that, let us estimate the size of the intersection of D with the affine hyperplane $\sum_{i=0}^{n-1} a_i x_i = 0$.

Since the inner product $\langle\cdot,\cdot\rangle$ is bilinear over $\mathbf{Z}_2$ we have

$$\sum_{i=0}^{n-1} a_i \langle u^i, v\rangle = \langle \sum_{i=0}^{n-1} a_i u^i, v\rangle.$$

Let $q_a(X)$ denote the polynomial $\sum_{i=0}^{n-1} a_i X^i \in \mathbf{F}[X]$. If u is a root of the polynomial $q_a(X)$, then clearly the inner product $\langle \sum_{i=0}^{n-1} a_i u^i, v\rangle$ is equal to 0. Now suppose $u \in \mathbf{F}$ is not a root of $q_a(X)$, then $\sum_{i=0}^{n-1} a_i u^i = q_a(u)$ is a non-zero element in $\mathbf{F}$. It is easy to see that for any non-zero $w \in \mathbf{F}$,

$$\mathbf{Pr}_{v\in\mathbf{F}}[\langle w, v\rangle = 0] = 1/2.$$

Thus,

$$\begin{aligned} &\mathbf{Pr}_{u,v\in\mathbf{F}}\left[\sum_{i=0}^{n-1} a_i\langle u^i, v\rangle = 0\right] \\ = \quad &\mathbf{Pr}_{u\in\mathbf{F}}[u \text{ is a root of } q_a(X)] + \mathbf{Pr}_{u\in\mathbf{F}}[u \text{ is not a root of } q_a(X)] \cdot \frac{1}{2}. \end{aligned}$$

But $q_a(X)$ is a non-zero polynomial of degree at most $n-1$, thus

$$\mathbf{Pr}_{u\in\mathbf{F}}[u \text{ is a root of } q_a(X)] \leq \frac{n-1}{2^m}.$$

Collecting terms, we have

$$\left|\mathbf{Pr}_{u,v\in\mathbf{F}}\left[\sum_{i=0}^{n-1} a_i\langle u^i, v\rangle = 0\right] - \frac{1}{2}\right| \leq \frac{n-1}{2^{m+1}}.$$

In particular, if $m > \log_2 n$, both affine hyperplanes $\sum_{i=0}^{n-1} a_i x_i = 0$ and $\sum_{i=0}^{n-1} a_i x_i = 1$ must intersect our set D.

In general, consider any k linearly independent equations $\sum_{j=0}^{n-1} a_{ij} x_j = b_i$, where $a_{ij}, b_i \in \mathbf{Z}_2$, and $i = 1, \ldots, k$. Denote this affine space by Π. Denote the point in D specified by u, v as $D(u, v)$. Generalizing the ideas above, we can show that

$$\left|\mathbf{Pr}_{u,v\in\mathbf{F}}[D(u,v) \in \Pi] - \frac{1}{2^k}\right| < \frac{n}{2^{m-k}},$$

which by our choice of m and k is bounded above by $1/2^{k+1}$. Thus, in particular,

$$\mathbf{Pr}_{u,v\in\mathbf{F}}[D(u,v) \in \Pi] > 0.$$

Other than linear independence, the coefficient vectors and the vector $(b_1, \ldots, b_k)$ on the right hand side in the definition of Π are arbitrary; the total number of the b vectors is $2^k = 2p(n) > p(n)$, and it follows that no linear subspace M of dimension $< n - \log_2 p(n)$ can cover the set D with some $p(n)$ displacements.

Theorem 5.2 *If there is a sparse set S which is hard for* P *under* NC^2 *many-one reductions, then* $\mathrm{P} = \mathrm{NC}^2$.

5.3 *The Finale:* NC^1 *Simulation*

The ideas discussed above paved way for the final assault—an optimal simulation in NC^1, circuits of polynomial size and logarithmic depth.

This is really quite fortunate. Note that the proof ideas so far are basically to form a certain linear equation system and solve it. After some ten years of continued progress, starting with the pioneer work of Csanky, culminating in the remarkable achievement of Mulmuley, it was shown ultimately that many tasks in linear algebra such as rank, determinant, inverse, etc. *over arbitrary characteristic* can be done in NC^2 [Csa76, BvzGH82, Chi85, Mul87]. There is no evidence that all these heavy duty computations can be accomplished in NC^1. On the contrary, NC^1 is considered rather weak; it was the celebrated AKS sorting network [AKS83] that finally puts `sorting` in NC^1. It appears that it would be overly optimistic to hope to do all that linear algebra in NC^1.

But Cai and Sivakumar succeeded in proving the following strong result:

Theorem 5.3 *If a sparse set S is hard for* P *under many-one reductions, then the* P*-complete circuit-value problem can be solved by a logspace-uniform family of polynomial size, logarithmic depth circuits that make polynomially many parallel calls to the reduction.*

That is, modulo the complexity of the reduction to the sparse set, the resulting algorithm can be implemented by a uniform NC^1 circuit of polynomial size and logarithmic depth. It follows that if the reduction itself is computable in logspace-uniform NC^1, then P equals logspace-uniform NC^1; and, for the Hartmanis conjecture, if there is a logspace computable many-one reduction from P to a sparse set S, then P = LOGSPACE.

Proof It is known that the polynomial $X^{2\cdot 3^\ell} + X^{3^\ell} + 1 \in \mathbf{Z}_2[X]$ is an irreducible polynomial over $\mathbf{Z}_2$ for all $\ell \geq 0$ [vL91]. In the following, by a finite field $GF(2^m)$, where $m = 2 \cdot 3^\ell$, we refer explicitly to the field $\mathbf{Z}_2[X]/(X^{2\cdot 3^\ell} + X^{3^\ell} + 1)$.

Let S be a sparse set hard for P under logspace-computable many-one reductions. As before, we will consider a refinement of the circuit-value problem. Define

$$L = \left\{ \langle C, x, 1^m, u, v \rangle \mid m = 2 \cdot 3^\ell, u, v \in GF(2^m), \sum_{i=0}^{n-1} u^i g_i = v \right\},$$

where C is a boolean circuit and x is an input to C, and where $g_0, \ldots, g_{n-1}$ are 0-1 variables that denote the values of the gates of C on input x. Here exponentiation and summation are carried out in the finite field $GF(2^m)$. It is easy to see that $L \in \mathrm{P}$, since all the required field arithmetic involved in checking $\sum u^i g_i = v$ can be performed in polynomial time.

Clearly $|\langle C, x, 1^m, u, v \rangle|$ is bounded polynomially in n and m. If f is a logspace-computable function that reduces L to S, the bound on the length of queries made by f on inputs of length $|\langle C, x, 1^m, u, v \rangle|$ is some polynomial $q(n, m)$. Let $p(n, m)$ be a polynomial that bounds the number of strings in S of length at most $q(n, m)$. We will choose the smallest m of the form $2 \cdot 3^\ell$ such that $2^m / p(n, m) \geq n$. It is clear that $m = O(\log n)$. Let $\mathbf{F}$ denote the finite extension $GF(2^m)$ of $GF(2)$.

Facts. We first collect some facts about implementing the basic operations of $\mathbf{F}$. For each operation, the number of processors needed is at most $n^{O(1)}$.

(1) Finding a primitive element ω that generates the multiplicative group $\mathbf{F}^*$ of $\mathbf{F}$ can be done in logspace by exhaustive search.

(2) Adding two elements $y_1, y_2 \in \mathbf{F}$ is just the bitwise exclusive-or of the representations of y_1 and y_2, and can be done in depth $O(1)$. Adding $n^{O(1)}$-many elements can be done in depth $O(\log n)$.

(3) Using logarithmic space, it is also possible to build the multiplication table for $\mathbf{F}$, so multiplying two elements of $\mathbf{F}$ can be done by a circuit of depth $O(\log \log n)$ and size $(\log n)^{O(1)}$.

(4) Raising the generator ω to any power $i < 2^m$, or computing the discrete logarithm of any element with respect to ω, can be done by table lookup in depth $O(\log \log n)$. The tables themselves can be precomputed using $O(\log n)$ space.

(5) Multiplying $k = n^{O(1)}$ elements of $\mathbf{F}$ can be done in $O(\log n)$ depth. The idea is to use the discrete logarithms of the k elements with respect to the generator ω, and convert multiplications to additions of k $O(\log n)$-bit integers (modulo $2^m - 1$), which can be done in $O(\log n)$ depth using the folklore 3-to-2 trick.

Our parallel algorithm for CVP begins by computing $f(\langle C, x, u, v \rangle)$ for all $u, v \in \mathbf{F}$. For every non-zero $u \in \mathbf{F}$, there is a unique element $v_u \in \mathbf{F}$ such that $\langle C, x, u, v_u \rangle \in L$, and therefore $f(\langle C, x, u, v_u \rangle) \in S$. Since $2^m / p(n, m) \geq n$, there is at least one string $w \in S$ such that the number of u satisfying $f(\langle C, x, u, v_u\rangle) = w$ is at least n. Of course, there could be many such w (not necessarily in S), and we don't know which w is a string in S. To handle this, we will assume that every w that has $\geq n$ preimages is a string in S, and attempt to solve for the g_i's. As long as there is at least one $w \in S$ that has $\geq n$ preimages, one of the assumptions must be correct, and we will have the correct solution. Since we know the details of the circuit C, the solutions can be verified, and the incorrect ones weeded out.

Assume, therefore, without loss of generality, that $w \in S$ has $\geq n$ preimages. Let $u_1, u_2, \ldots, u_n$ denote n of them, and let $v_1, v_2, \ldots, v_n$ denote the corresponding v_u's. The equations

$$1g_0 + u_j g_1 + u_j^2 g_2 + \ldots + u_j^{n-1} g_{n-1} = v_j, \qquad j = 1, 2, \ldots, n$$

form an inhomogeneous system of linear equations, where the coefficients u_j^i form a *Vandermonde* matrix. Since the u_j's are distinct elements of the field $\mathbf{F}$, the system has full rank.

It remains to show how to solve this system of equations in NC^1. While solving general linear equation systems seems to require NC^2, we will arrive at our NC^1 solution via closed formulae.

Using properties of the Vandermonde system one can show the following closed formula:

$$g_j = \sum_{i=1}^{n} (-1)^{1+i} \frac{v_i}{\prod_{k \neq i}(u_k - u_i)} P_{n-j-1}(u_1, \ldots, \widehat{u_i}, \ldots, u_n),$$

for $j = 1, 2, \ldots, n$. Here $\widehat{u_i}$ denotes that u_i is missing from the list $u_1, \ldots, u_n$, and P_k denotes the k-th elementary symmetric polynomial, defined as follows:

$$P_0(y_1, \ldots, y_\ell) = 1; \qquad P_k(y_1, \ldots, y_\ell) = \sum_{\substack{I \subseteq [\ell] \\ \|I\| = k}} \prod_{i \in I} y_i, \qquad k > 0.$$

By Facts (3) and (5), computing $v_i / (\prod_{k \neq i}(u_k - u_i))$ in NC^1 is fairly straightforward. Hence it suffices to show that the polynomials $P_k(u_1, \ldots, \widehat{u_i}, \ldots, u_n)$ are computable in logspace-uniform NC^1. A folklore theorem indicates that this can be done in non-uniform NC^1. For our application, however, the uniformity is crucial.

It is easy to see that for any $y_1, \ldots, y_\ell \in \mathbf{F}$, $P_k(y_1, \ldots, y_\ell)$ is equal to $P_k(y_1, y_2, \ldots, y_\ell, 0, 0, \ldots, 0)$ for any number of extra zeroes. Let $r = \|\mathbf{F}^*\|$, the number of elements in the multiplicative group of $\mathbf{F}$. We will give an NC^1 algorithm to compute the elementary symmetric polynomials of r elements, not necessarily distinct, from the finite field $\mathbf{F}$. By appending $r - \ell$ zeroes, we can then compute $P_k(y_1, y_2, \ldots, y_\ell)$.

For each $k, 0 < k \leq r$, the value of the elementary symmetric polynomial $P_k(y_1, y_2, \ldots, y_r)$ is the coefficient of X^{r-k} in $h(X) \stackrel{\text{def}}{=} \prod_{i=1}^{r}(X + y_i) - X^r$.

Note that, given any $\alpha \in \mathbf{F}$, $h(\alpha)$ can be evaluated in NC^1, by Facts (2) and (5).

If we write $h(X)$ as $\sum_{i=0}^{r-1} a_i X^i$, the coefficient $a_i = P_{r-i}(y_1, \ldots, y_r)$ for $0 \le i < r$. The idea now is to choose α's carefully from $\mathbf{F}$, compute $h(\alpha)$ and compute the coefficients a_i by interpolation. If we choose ω to be a primitive element of order r in $\mathbf{F}^*$, the powers of ω, namely $1 = \omega^0, \omega^1, \omega^2, \ldots, \omega^{r-1}$, run through the elements of $\mathbf{F}^*$. For $0 \le i < r$, let $b_i = h(\omega^i)$. The relationship between the pointwise values (b_i's) and the coefficients (a_i's) of $h(X)$ can be written as:

$$\begin{pmatrix} b_0 \\ b_1 \\ \vdots \\ b_{r-1} \end{pmatrix} = \begin{pmatrix} 1 & \omega^0 & \omega^{0\cdot 2} & \ldots & \omega^{0\cdot(r-1)} \\ 1 & \omega^1 & \omega^{1\cdot 2} & \ldots & \omega^{1\cdot(r-1)} \\ \vdots & \vdots & \vdots & \ldots & \vdots \\ 1 & \omega^{r-1} & \omega^{(r-1)\cdot 2} & \ldots & \omega^{(r-1)\cdot(r-1)} \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_{r-1} \end{pmatrix}$$

The matrix Ω is the Discrete Fourier Transform matrix, and is a Vandermonde matrix. Since the powers of ω are all distinct, Ω is invertible, and one can compute the coefficients a_i by $(a_0, \ldots, a_{r-1})^T = \Omega^{-1}(b_0, \ldots, b_{r-1})^T$. The crucial advantage over the earlier Vandermonde system is that with this particular choice of Ω, the matrix Ω^{-1} has a simple explicit form:

$$\Omega_{ij}^{-1} = 1/(\Omega_{ij}) = \omega^{-ij}.$$

Computing the coefficients of $h(X)$ is now simply a matrix-vector multiplication. Theorem 5.3 is proven. □

Corollary 5.1 *If there is a sparse set S that is hard for* P *under logspace-computable many-one reductions, then* P = LOGSPACE.

Corollary 5.2 *If there is a sparse set S that is hard for* P *under many-one reductions computable in logspace-uniform* NC^1*, then* P *equals logspace-uniform* NC^1*.*

The algebraic approach presented here is rather powerful. Many additional results have been obtained including the Hartmanis conjecture for NL, probabilistic and other weaker reductions, and logspace bounded truth table reductions for P [CS95b, CNS95, vM95].

6 Conclusions

We have seen that the study of sparse sets shows great vitality. Significant progress is being made which provides unifying perspectives to known and new results, and reaches out to new territory with unexpected links to parallel computation and randomization. In particular, the proof on the

Hartmanis Conjecture for P has reached a new level of sophistication, one might wonder if there is a more elementary proof.

In the last ten years, there is a growing realization that a source of randomness can be effectively viewed as a computational resource. Whether this view is a Platonic truth, i.e., whether randomness in computation cannot be ultimately eliminated without any loss in efficiency, is unresolved. The philosophical point is both fascinating and troubling. One might reasonably ask:

> Why couldn't one find deterministically a solution to a problem, which has nothing probabilistic in its definition, as efficiently as a probabilistic procedure would?

Despite this philosophical reservation, the enormous power and effectiveness *using* randomization cannot be denied. Solutions that are otherwise unavailable are frequently found with the help of randomness. Sometimes the use of randomness can be eliminated later, sometimes not. And even when one can eventually eliminate randomness, frequently one does that via a process called derandomization whereby a pseudorandom source, which is generated deterministically, is substituted for a true random source, thus "mimicking" randomness. To paraphrase Eugene Wigner, there is overwhelming empirical evidence for *the unreasonable effectiveness of a coin toss in computation*, which leaves no doubt that randomization is an extremely valuable tool in algorithm design and complexity theory.

The impact of randomization in structural complexity theory in the past has not been perhaps as decisive as in some other areas of theoretical computer science. It is hoped that the recent work on sparse sets will help change that perception. We believe that randomization will be increasingly important in structural complexity theory in the future.

7 References

[AA95] M. Agrawal and V. Arvind. Reductions of self-reducible sets to depth-1 weighted threshold circuit classes, and sparse sets. In *Proc. 10th Conf. on Structure in Complex. Theory*, pages 264–276. IEEE, June 1995.

[AHH+93] V. Arvind, Y. Han, L. Hemachandra, J. Köbler, A. Lozano, M. Mundhenk, M. Ogiwara, U. Schöning, R. Silvestri, and T. Thierauf. Reductions to sets of low information content. In K. Ambos-spies, S. Homer, and U. Schöning, editors, *Complexity Theory*, pages 1–46. Cambridge University Press, 1993.

[AKM92a] V. Arvind, J. Köbler, and M. Mundhenk. Lowness and the complexity of sparse and tally descriptions. In *Proc. 3rd Int'l Symp. on Alg. and Comput.*, pages 249–258. Springer-Verlag Lecture Notes in Comp. Sci. *#650*, December 1992.

[AKM92b] V. Arvind, J. Köbler, and M. Mundhenk. On bounded truth-table, conjunctive, and randomized reductions to sparse sets. In *Proc. 12th Symp. on Found. of Software Tech. and Theoret. Comp. Sci.*, pages 140–151. Springer-Verlag Lecture Notes in Comp. Sci. *#652*, 1992.

[AKS83] M. Ajtai, J. Komlos, and E. Szemeredi. Sorting in $c \log n$ parallel steps. *Combinatorica*, 3(1):1–19, 1983.

[BCG+95] N. Bshouty, R. Cleve, R. Gavaldà, S. Kannan, and C. Tamon. Oracles and queries that are sufficient for exact learning. Tech. Report TR95-015, The Electronic Colloquium on Computational Complexity, 1995.

[BDG88] J. Balcázar, J. Díaz, and J. Gabarró. *Structural Complexity I.* EATCS Monographs in Theoretical Computer Science. Springer-Verlag, 1988.

[Ber78] P. Berman. Relationship between density and deterministic complexity of NP-complete languages. In *Proc. 5th Conf. on Automata, Lang. and Prog.*, pages 63–71. Springer-Verlag Lecture Notes in Comp. Sci. *#62*, 1978.

[BH77] L. Berman and J. Hartmanis. On isomorphisms and density of NP and other complete sets. *SIAM J. Comput.*, 6(2):305–322, 1977.

[BvzGH82] A. Borodin, J. von zur Gathen, and J. Hopcroft. Fast parallel matrix and GCD computations. *Inf. and Control*, 52:241–256, 1982.

[Chi85] A. Chistov. Fast parallel calculation of the rank of matrices over a field of arbitrary characteristic. In *Proc. 5th Int'l Conf. on Fundamentals of Comput. Theory*, pages 63–69. Springer-Verlag Lecture Notes in Comp. Sci. *199*, 1985.

[CNS95] J. Cai, A. Naik, and D. Sivakumar. On the existence of hard sparse sets under weak reductions. Tech. Report 95-31, Department of Computer Science, State University of New York at Buffalo, Buffalo, NY, July 1995.

[Cob65] A. Cobham. The intrinsic computational difficulty of functions. In Y. Bar-Hellel, editor, *Proc. 1964 Int'l Cong. for Logic, Methodology, and Phil. of Sci.*, pages 24–30. North Holland, Amsterdam, 1965.

[Coo71] S. Cook. The complexity of theorem proving procedures. In *Proc. 3rd Symp. on Theory of Comput.*, pages 151–158. ACM Press, 1971.

[CS95a] J. Cai and D. Sivakumar. The resolution of a Hartmanis conjecture. In *Proc. 36th Symp. on Found. of Comp. Sci.* IEEE, 1995. To appear.

[CS95b] J. Cai and D. Sivakumar. The resolution of a Hartmanis conjecture for NL. Manuscript, 1995.

[Csa76] L. Csanky. Fast parallel matrix inversion algorithms. *SIAM J. Comput.*, 5(4):618–623, 1976.

[FFK92] S. Fenner, L. Fortnow, and S. Kurtz. The isomorphism conjecture holds relative to an oracle. In *Proc. 33rd Symp. on Found. of Comp. Sci.*, pages 30–39. IEEE, October 1992.

[For79] S. Fortune. A note on sparse complete sets. *SIAM J. Comput.*, 8(3):431–433, 1979.

[GJ79] M. Garey and D. Johnson. *Computers and intractability: A guide to the theory of NP-completeness.* W. H. Freeman and Company, 1979.

[GNW90] T. Gundermann, N. Nasser, and G. Wechsung. A survey of counting classes. In *Proc. 5th Conf. on Structure in Complex. Theory*, pages 140–153. IEEE, July 1990.

[GS88] J. Grollmann and A. Selman. Complexity measures for public-key cryptosystems. *SIAM J. Comput.*, 17(2):309–335, 1988.

[Har78] J. Hartmanis. On log-tape isomorphisms of complete sets. *Theoret. Comput. Sci.*, 7(3):273–286, 1978.

[Hel86] H. Heller. On relativized exponential and probabilistic complexity classes. *Inf. and Control*, 71(3):231–243, 1986.

[HL94] S. Homer and L. Longpré. On reductions of NP sets to sparse sets. *J. Comput. System Sci.*, 48(2):324–336, 1994.

[HOW92] L. Hemachandra, M. Ogiwara, and O. Watanabe. How hard are sparse sets. In *Proc. 7th Conf. on Structure in Complex. Theory*, pages 222–238. IEEE, June 1992.

[HS65] J. Hartmanis and R. Stearns. On the computational complexity of algorithms. *Trans. Amer. Math. Soc.*, 117:285–306, 1965.

[IM89] N. Immerman and S. Mahaney. Relativizing relativized computations. *Theoret. Comput. Sci.*, 68(3):267–276, 1989.

[JY85] D. Joseph and P. Young. Some remarks on witness functions for non-polynomial and non-complete sets in NP. *Theoret. Comput. Sci.*, 39:225–237, 1985.

[Kad89] J. Kadin. $\mathrm{P}^{\mathrm{NP}[\log n]}$ and sparse Turing-complete sets for NP. *J. Comput. System Sci.*, 39(3):282–298, 1989.

[Kar] R. Karp. Personal communication.

[KL82] R. Karp and R. Lipton. Turing machines that take advice. *L'enseignement Mathematique*, 28(3/4):191–209, 1982.

[KMR90] S. Kurtz, S. Mahaney, and J. Royer. The structure of complete degrees. In A. Selman, editor, *Complexity Theory Retrospective*, pages 108–146. Springer-Verlag, 1990.

[KW95] J. Köbler and O. Watanabe. New collapse consequences of NP having small circuits. In *Proc. 22nd Conf. on Automata, Lang. and Prog.*, pages 196–207. Springer-Verlag Lecture Notes in Comp. Sci. *#944*, July 1995.

[Lad75] R. Ladner. The circuit value problem is log space complete for P. *SIGACT News*, 7(1):18–20, 1975.

[Lon82] T. Long. A note on sparse oracles for NP. *J. Comput. System Sci.*, 24:224–232, 1982.

[Mah82] S. Mahaney. Sparse complete sets for NP: Solution of a conjecture of Berman and Hartmanis. *J. Comput. System Sci.*, 25(2):130–143, 1982.

[Mah89] S. Mahaney. The isomorphism conjecture and sparse sets. In J. Hartmanis, editor, *Computational Complexity Theory*, pages 18–46. American Mathematical Society Proceedings of Symposia in Applied Mathematics #38, 1989.

[Mil76] G. Miller. Riemann's hypothesis and tests for primality. *J. Comput. System Sci.*, 13:300–317, 1976.

[Mul87] K. Mulmuley. A fast parallel algorithm to computer the rank of a matrix over an arbitrary field. *Combinatorica*, 7(1):101–104, 1987.

[Ogi95] M. Ogihara. Sparse hard sets for P yield space-efficient algorithms. In *Proc. 36th Symp. on Found. of Comp. Sci.* IEEE, 1995. To appear.

[OL93] M. Ogiwara and A. Lozano. Sparse hard sets for counting classes. *Theoret. Comput. Sci.*, 112(2):255–276, 1993.

[OW91] M. Ogiwara and O. Watanabe. On polynomial time bounded truth-table reducibility of NP sets to sparse sets. *SIAM J. Comput.*, 20(3):471–483, 1991.

[Pip79] N. Pippenger. On simultaneous resource bound. In *Proc. 11th Symp. on Theory of Comput.*, pages 307–311. ACM Press, 1979.

[Rab59] M. Rabin. Speed of computation and classification of recursive sets. In *Proc. 3rd Convention Sci. Soc.*, pages 1–2, 1959.

[RR92] D. Ranjan and P. Rohatgi. On randomized reductions to sparse sets. In *Proc. 7th Conf. on Structure in Complex. Theory*, pages 239–242. IEEE, June 1992.

[Ukk83] E. Ukkonen. Two results on polynomial time truth-table reductions to sparse sets. *SIAM J. Comput.*, 12(3):580–587, 1983.

[vL91] J. van Lint. *Introduction to Coding Theory.* Springer-Verlag, 1991.

[vM95] D. van Melkebeek. On truth-table reductions of P sets to sparse sets. Manuscript, 1995.

[Wat92] O. Watanabe. On polynomial-time one-truth-table reducibility to a sparse set. *J. Comput. System Sci.*, 44:500–516, 1992.

[Yam62] H. Yamada. Real time computation and recursive functions not real-time computable. *IRE Trans. Electronics Computers*, EC-11:753–760, 1962.

[Yap83] C. Yap. Some consequences of non-uniform conditions on uniform classes. *Theoret. Comput. Sci.*, 26(3):287–300, 1983.

[Yes83] Y. Yesha. On certain polynomial-time truth-table reducibilities of complete sets to sparse sets. *SIAM J. Comput.*, 12(3):411–425, 1983.

[You90] P. Young. Juris Hartmanis: fundamental contributions to isomorphism problems. In A. Selman, editor, *Complexity Theory Retrospective*, pages 28–58. Springer-Verlag, 1990.

[You92] P. Young. How reductions to sparse sets collapse the polynomial-time hierarchy: A primer. *SIGACT News*, 1992. Part I (#3, pages 107–117), Part II (#4, pages 83–94), and Corrigendum (#4, page 94).