Sparse Hard Sets for P

Dieter van Melkebeek
Department of Computer Science
University of Chicago
Chicago, IL 60637

Mitsunori Ogihara
Department of Computer Science
University of Rochester
Rochester, NY 14627

Abstract

Sparse hard sets for complexity classes has been a central topic for two decades. The area is motivated by the desire to clarify relationships between completeness/hardness and density of languages and studies the existence of sparse complete/hard sets for various complexity classes under various reducibilities. Very recently, we have seen remarkable progress in this area for low-level complexity classes. In particular, the Hartmanis' sparseness conjectures for P and NL have been resolved. This article overviews the history of sparse hard set problems and exposes some of the recent results.

1 INTRODUCTION

1.1 The Birth of the Area

The most important discovery in computational complexity theory is no doubt NP-completeness. Since Cook [Coo71] and Levin [Lev73] independently established the notion, thousands of natural NP-complete problems have been identified in various scientific fields (for a list of NP-complete problems, see e.g., [GJ79]). NP-complete problems are a "core" of NP in the sense that any of them belonging to NP implies P=NP. Together with the fact that none of them are known to be in P, these numerous NP-complete problems seem to amount to evidence that $\mathrm{P} \neq \mathrm{NP}$. Also, this huge list of NP-complete problems gives rise to the question what intrinsic properties NP-complete sets possess: Are there any structural properties other than being complete, that NP-complete sets have in common?

In 1976, Berman and Hartmanis suggested such a property [BH77]. The two researchers examined all NP-complete sets known at the time and showed that they are p-isomorphic, i.e., isomorphic with respect to polynomial time computable, polynomial time invertible bijections. Based on this observation, they conjectured that the isomorphism property should hold for *all* NP-complete sets; in other words, there is only one $\leq_m^p$-complete degree in NP (the isomorphism conjecture). Intuitively, this means that an arbitrary NP-complete set can be generated from any other NP-complete set by means of efficient renaming procedures of strings, and thus, there is basically only one, unique NP-complete set. This simple, plausible statement has attracted many brilliant scientists over the past two decades and has become a central research area in complexity theory. (The reader may refer to [KMR90, You90, FFK92] for the current status of the isomorphism conjecture.)

What does this conjecture suggest? First of all, $\mathrm{P} \neq \mathrm{NP}$, because finite sets are in $\mathrm{P} \subseteq \mathrm{NP}$ but not isomorphic to SAT. Second, more amusingly, sparse sets are not NP-complete. For a language A, define the census function of A, $cens_A(n)$, to be the one that maps each n to the number of elements in A of length up to n. The set A is sparse if its census function is bounded by a polynomial. A typical NP-complete set such as SAT is dense—its census function is approximately of the form $a2^{cn}$. In addition, all polynomial time computable bijections, in particular p-isomorphisms, are density-preserving. So, the isomorphism conjecture implies that only dense sets are NP-complete, and therefore, sparse sets are not NP-complete. The latter observation led the two researchers to make the following conjecture.

Conjecture 1 The Sparseness Conjecture for NP

No sparse set is NP-complete.

The area of sparse hard sets was born as the study of the sparseness conjecture. In general, it investigates the existence of sparse complete/hard sets for various complexity classes with respect to various reducibilities. Apart from its relevance to the isomorphism conjecture discussed above, a major goal is to obtain connections between Turing machine based complexity and circuit based complexity, in other words, uniform complexity measures and nonuniform ones. Berman and Hartmanis, in their conjecture paper, observed that the $\leq_T^p$ reducibility closure of the sparse sets is identical to "nonuniform P" (the class of sets recognized by polynomial size circuits), denoted by $\mathrm{P/poly}$ [KL80]. The property of a class $\mathcal{C}$ being reducible to sparse sets then can be viewed as that of possessing small nonuniform complexity. This relation is interesting, because it provides fine classifications of $\mathrm{P/poly}$: According to the power of the access, the reducibility closures of the sparse sets form proper hierarchies within $\mathrm{P/poly}$ (for such results, see [BK88, Ko89, AHOW92]). Thus, by-products of the results on sparse hard set problems are nonuniform lower bounds of uniform complexity classes, from a totally different angle than usual circuit complexity lower bound arguments.

1.2 Progress on the Sparseness Conjecture for NP

The sparseness conjecture by Berman and Hartmanis triggered a line of research leading to the resolution of the conjecture. In 1980, Karp and Lipton [KL80] showed that NP, PSPACE, $\mathrm{EXPTIME} = \bigcup_{k>0} 2^{n^k}$, and, in general, classes $\mathcal{C}$ with self-reducible complete sets, lack sparse $\leq_T^p$-hard sets unless $\mathrm{PH} = \Sigma_2^p$, $\mathrm{PSPACE} = \Sigma_2^p$, $\mathrm{EXPTIME} = \Sigma_2^p$, and $\mathrm{NP}^{\mathcal{C}} = \mathrm{coNP}^{\mathcal{C}}$, respectively. The result has set a lower bound on the strength of the collapse one could obtain assuming the existence of sparse hard sets.

On the other hand, for the $\leq_m^p$-case, Berman [Ber78] showed in 1978 that tally NP-hard sets would collapse NP to P, where tally sets are languages over one letter alphabets. A year later, Fortune [For79] improved this and showed that sparse coNP-hard sets would collapse NP to P. Both Berman and Fortune develop a polynomial time decision procedure for SAT under the assumption that SAT^c is $\leq_m^p$-reducible to some low density set. Their procedures both run depth first search on the self-reduction tree of SAT where the tree is cleverly pruned by comparing strings in a small list of images of the formulas appearing as node labels under the many-one reduction. The final piece of the puzzle was placed by Mahaney in 1980 [Mah82], who showed that sparse $\leq_m^p$-hard sets for NP exist if and only if $\mathrm{P} = \mathrm{NP}$. The key idea in Mahaney's proof is the introduction of the "pseudo-complement" of sparse complete sets. If $cens_S(n)$ of a sparse complete set S is known, then the membership test of a string x of length at most n in S^c is doable in NP: Find, by nondeterministic guesses, all the members in S of length up to n, then check that x is not in the member-list. Since S is NP-complete, this indicates S^c is "pseudo" reducible to S. Using a known polynomial bound on the census function of S, by trying all possible values of the census to create the pseudo-complement of S and running Fortune's method with respect to the pseudo-complement, a satisfying assignment of the input formula can be discovered if one exists. If S is only known to be NP-hard, we can apply the above arguments to the set $S' = \{0^{|x|}1f(x) \mid x \in \mathrm{SAT}\}$, where f is a $\leq_m^p$-reduction from SAT to S. The set S' is a sparse NP-complete set.

Mahaney's result left us with the question whether the reducibility can be strengthened to $\leq_{btt}^{p}$ while preserving the collapse. A set A is $\leq_{btt}^{p}$-reducible to a set B if there is a polynomial time bounded oracle Turing machine accepting A with B as the oracle, which makes at most k nonadaptive queries for some fixed constant k. It was not until 1990 that this question was answered. Ogihara and Watanabe introduced the concept of *left-sets* and proved that sparse $\leq_{btt}^{p}$-hard sets for NP would collapse NP to P [OW91].

Furthermore, Ogihara and Lozano generalized the notion and proved that the collapse $\mathcal{C} = \mathrm{P}$ follows from the existence of sparse $\leq_{btt}^{p}$-hard sets for $\mathcal{C}$ for many complexity classes above P [OL93]. These two breakthrough results started a "gold rush" in the study of sparse hard sets for classes above P and many interesting theorems were proven (for a survey, see [HOW92, You92, CO96]).

1.3 The Sparseness Conjecture for P and NL

The question whether there are sparse hard sets applies to other classes than NP as well, with appropriate choices of reducibilities. In 1978, Hartmanis studied the existence of sparse $\leq_{m}^{log}$-hard sets for various complexity classes and showed that PSPACE, $\mathrm{EXPTIME} = \bigcup_{c>0}\mathrm{Time}(2^{n^c})$, and $\mathrm{EXPSPACE} = \bigcup_{c>0}\mathrm{Space}(2^{n^c})$ all lack such sparse complete sets [Har78]. Then he insightfully conjectured that there are no sparse complete sets for P or NL under $\leq_{m}^{log}$-reductions.

Conjecture 2 The Sparseness Conjecture for P and NL

No sparse set is complete for P *or* NL.

As in the case of the sparseness conjecture for NP, one may expect to prove that the existence of such sparse sets would collapse these classes to seemingly smaller ones. However, until very recently, there has been little progress on the sparseness conjecture for P and NL. The one and only related observation in this regard is a result by Hemaspaandra, Ogihara, and Toda [HOT94] that polylog dense $\leq_{m}^{log}$-hard sets would collapse P (and NL) to Steve's class SC ($= \mathrm{TimeSpace}[\mathit{poly}, \mathit{polylog}]$) [Coo85]. The study has been stymied by the fact that the proof techniques for classes above P cannot be applied due to the lack of memory in logarithmic space. For example, both Mahaney and Ogihara-Watanabe, under the assumption that NP has sparse hard sets, develop a polynomial time search procedure for a satisfying assignment on the self-reduction tree of SAT. The procedure needs to maintain a polynomial number of nodes in the self-reduction tree of SAT. Clearly, logspace machines cannot do this.

In [Ogi95], for the first time in two decades, Ogihara demonstrated that one could overcome this difficulty. The key idea is the use of a language called $\mathrm{Parity\text{-}CVP}$, a variation of the circuit-value problem CVP, a typical P-complete problem [Lad75]. While CVP tests whether a given circuit C outputs 1 on a given input x, this auxiliary language tests, given an additional list of gates of C and an additional bit b, whether the output of the gates in the list adds up to b modulo 2. Parity-CVP is in P, so by hypothesis $\leq_m^{log}$-reduces to a sparse set. Using this reduction, Ogihara developed a procedure to translate the membership question for CVP to the problem of solving a system of linear equations over $\mathrm{GF}(2)$ in the outputs of the gates of C. The system is not full rank, leaving $O(\log n)$ indeterminate variables. By cycling through all possible assignments to these indeterminate variables and locally testing the correctness of the solution derived from the assignments, the output of all the gates can be determined. The space cost of the whole process is $O(\log^2 n)$, and thus, P having sparse $\leq_m^{log}$-hard sets implies $\mathrm{P} \subseteq \mathrm{Space}[\log^2 n]$.

Building upon this construction, Cai and Sivakumar obtained a resolution: P has sparse $\leq_m^{log}$-hard sets if and only if $\mathrm{P} = \mathrm{L}$ [CS95a]. The proof, which consists of three steps, is highly algebraic and full of clever ideas. The first step introduces randomness in Ogihara's reduction procedure. The randomization provides the collapse $\mathrm{P} \subseteq \mathrm{RNC}^2$, which is incomparable with $\mathrm{P} \subseteq \mathrm{Space}[\log^2 n]$. The second step consists of derandomizing the procedure in step 1 using small bias sample spaces, yielding the collapse $\mathrm{P} = \mathrm{NC}^2$. Now the result by Ogihara is a simple corollary. The final step introduces Reed-Solomon encodings of the gate values, instead of the Hadamard encodings used by Ogihara and in the first two steps of Cai-Sivakumar. This approach results in a Vandermonde system of linear equations, which can be solved in NC^1. This gives $\mathrm{P} \subseteq \mathrm{L}$ and Hartmanis' conjecture is resolved. The result has an added bonus: If the many-one reduction happens to be computable in NC^1, the collapse becomes $\mathrm{P} = \mathrm{NC}^1$. We will discuss the proof in more detail in the next section.

The proof technique by Cai and Sivakumar not only resolved the long standing open question for P, but also paved the way to settle the other conjecture by Hartmanis. First, Cai, Naik, and Sivakumar [CNS96] proved that sparse $\leq_m^{log}$-hard sets for NL collapse NL to randomized logspace. The collapse is then improved by Cai and Sivakumar [CS95b] to L.

Now that the conjectures are resolved, the Ogihara-Watanabe result motivates researchers to study whether the resolutions can be strengthened to bounded truth-table reductions. In this regard, Cai, Naik, and Sivakumar [CNS96] showed that $\mathrm{P} = \mathrm{NC}^2$ holds if sparse $\leq_{btt}^{log}$-hard sets exist for P. The collapse has been improved by Van Melkebeek [vM96] to $\mathrm{P} = \mathrm{L}$, so, the bounded truth-table case is completely solved.

The technique by Van Melkebeek can be applied to all complexity classes with simple unique membership proofs. He showed that sparse $\leq_{btt}^{log}$-hard sets for NL collapses NL to L. Furthermore, as in the case of the Cai-Sivakumar resolution for P, the reduction procedure is actually an NC^1 circuit with parallel queries to the reduction as a black box, so, the collapse can be strengthened to NC^1 if the bounded truth-table reduction is computed by NC^1-circuits.

1.4 Organization of the article

This article is organized as follows. Section 2 presents the breakthrough by Ogihara and the resolution by Cai and Sivakumar. Section 3 exposes the bounded truth-table reducibility proof by Van Melkebeek and presents how the technique can be applied to classes other than P. Section 4 discusses some future research topics.

2 SPARSE MANY-ONE HARD SETS FOR P: FROM OGIHARA TO CAI-SIVAKUMAR

A standard P-complete problem is the circuit value problem CVP [Lad75], which asks whether a given circuit outputs 1 on a given input. In his breakthrough paper, Ogihara introduced a variation of CVP, called $\mathrm{Parity\text{-}CVP}$. $\mathrm{Parity\text{-}CVP}$ is the collection of all quadruples $\langle C, x, I, b\rangle$ such that

- C is a circuit consisting of say n gates,
- x is an input to C,
- $I \in \{0,1\}^n$, $b \in \{0,1\}$, and
- $I \cdot \gamma = b$,

where γ is the vector $(g_1, \cdots, g_n)$ with g_i denoting the output of the i-th gate of C on x and $I \cdot \gamma$ is the inner product of I and γ over $\mathrm{GF}(2)$, the field of integers modulo 2.

This language $\mathrm{Parity\text{-}CVP}$ is in P, because for every i, g_i can be computed by asking CVP whether C on x outputs 1 with the i-th gate being the output gate. So, if we assume that P has a sparse $\leq_m^{log}$-hard set S, there is a $\leq_m^{log}$-reduction f of $\mathrm{Parity\text{-}CVP}$ to S. We will develop a method for computing γ and hence the output value of C on input x, which is more efficient than "polynomial time" methods. Then from the P-completeness of CVP, it will follow that P collapses to a smaller class. For simplicity, in the following discussion, let $\langle C, x\rangle$ be fixed.

Now we introduce the notion of "collision," which plays a crucial role. We say that $\langle C, x, I, b\rangle$ and $\langle C, x, J, c\rangle$ *collide* if both have the same image under f. Furthermore, we will say that I and J collide if for some choice of bits b and c, $\langle C, x, I, b\rangle$ and $\langle C, x, J, c\rangle$ collide. If I and J collide with bits b and c, then f being a many-one reduction of $\mathrm{Parity\text{-}CVP}$ implies

$$I \cdot \gamma = b \text{ if and only if } J \cdot \gamma = c.$$

Taking the modulo 2 sum of the two conditions yields

$$(I + J) \cdot \gamma = (b + c).$$

This is a linear equation in γ over the field Z_2. We call $(I + J)$ the *coefficient vector* of the collision pair and $(b+c)$ the *value* of the collision pair. Note that if I and J collide, the value of the collision pair is unique.

Our aim is to find many collision pairs and obtain a large collection of linearly independent equations from them. Suppose we have collected $m = n - t$ linearly independent equations $A_i \cdot \gamma = d_i, 1 \leq i \leq m$, where t is $O(\log n)$. The system of linear equations we have obtained can be written as:

$$A \cdot \gamma = d. \tag{1}$$

Since the equations are linearly independent, the rank of the matrix A is m. By splitting A into two parts we obtain the following equation:

$$A_1 \cdot \gamma_1 = d - A_2 \cdot \gamma_2, \tag{2}$$

where A_1 is a full rank $m \times m$ matrix and A_2 is a $t \times t$ matrix.

For every $\gamma_2 \in \{0, 1\}^t$, we can solve the system (2) to obtain γ_1, and locally check the resulting γ for consistency:

- if g_i computes the AND (respectively, OR) of g_j and g_k, then $g_i = g_j \wedge g_k$ (respectively, $g_i = g_j \vee g_k$);
- if g_i computes the negation of g_j, then $g_i = 1 - g_j$; and
- if g_i is an input gate attached to the j-th input bit, then $g_i = x_j$.

The correct value of γ_2 will result in the correct value of γ, and only that vector will pass all these tests.

For a moment, let us postpone the discussion of how to collect collision pairs and construct the system of linear equations, and evaluate how much resources we need for the rest. First, matrix-vector multiplication over $\mathrm{GF}(2)$ is in NC^1 and solving full-rank systems of linear equations over $\mathrm{GF}(2)$ can be done in NC^2 [BvzGH82]. So, for any fixed $\gamma_2 \in \{0,1\}^t$, we can compute γ_1 in NC^2. Second, the consistency test for a fixed assignment to γ_2 can be done by a uniform $O(\log n)$ depth circuit. There are 2^t possible assignments for γ_1 and we have to select the correct γ based on the results of the consistency test. Cycling through the 2^t possible values for γ_1, and selecting the correct γ can be done in NC^1, provided $t \in O(\log n)$. Hence, the resulting algorithm requires NC^2 computation.

2.1 Ogihara's approach

Now we turn to the problem of constructing a system of linear equations of rank $n-t$ with the "deficiency" $t \in O(\log n)$. This is where the sparseness of S comes into play. Let S_0 denote the set of all $y \in S$ such that $y = f(\langle C, x, I, b\rangle)$ for some I and b. Since C can be encoded as an $O(n^2)$ bit string and S is sparse, there is a fixed polynomial p that bounds $|S_0|$. The strings in S_0 may appear many times. Since f many-one reduces $\mathrm{Parity\text{-}CVP}$ to S, for every I, exactly one of $f(\langle C, x, I, 0\rangle)$ and $f(\langle C, x, I, 1\rangle)$ belongs to S. So, the above bound suggests that there is a collision pair in every set of at least $p(n)+1$ many vectors. Based on this observation, Ogihara devised a search method for obtaining a system of high rank. Let $\mathcal{F}$ be the set of all n-bit vectors with at most $\lceil \log p(n) \rceil$ 1s. For each i, $1 \le i \le n$, search for a pair $(I, J) \in \mathcal{F}^2$ such that $I + J$ has 1 at the i-th entry and 0 at the j-th entry for all $j > i$. If such a pair exists, then pick the smallest one in lexicographic order to generate a linear equation. All these equations are linearly independent. By a counting argument, Ogihara showed that there are at most $\lceil \log p(n) \rceil$ many i for which the search fails. Thus, the deficiency t of the resulting system is $O(\log n)$. We need $O(\log^2 n)$ bits to encode each $I \in \mathcal{F}$ as an enumeration in binary of the indices of the 1-entries. So, a deterministic $O(\log^2 n)$ space algorithm can find a system of linear equations with the desired rank. Since $\mathrm{NC}^2 \subseteq \mathrm{Space}[\log^2 n]$, the overall complexity becomes $\mathrm{Space}[\log^2 n]$.

2.2 Collapsing to RNC^2

Ogihara's method, though correct, is not very efficient. We know that there is a collision pair in every $\mathcal{F}$ of size at least $p(n)+1$. Since p is a polynomial and we need at most n many linearly independent coefficient vectors, one might expect that for some large polynomial q, $q(n)$ many random vectors would yield a system of linear equations of high rank with high probability. Cai and Sivakumar showed this is indeed the case: For at least half of the families of $q(n) = 2(p(n)+1)n^2$ many n-bit vectors, the induced system of linear equations has rank $n - \log(p(n)+1)$. This observation suggests the following algorithm:

1. Select $\mathcal{F}$ by picking $q = 2(p(n)+1)n^2$ many n-bit vectors $I_1, \cdots, I_q$ uniformly at random.

2. Let $\mathcal{G}$ be the set of all non-identical pairs (i,j) from $\{1, \ldots, q\}$ such that I_i and I_j collide. Compute the rank m of the matrix constructed by collecting as rows all vectors $I_i + I_j$ with $(i,j) \in \mathcal{G}$. If m is less than $n - \log(p(n)+1)$, then assert "failure" and stop.

3. Consider the order over pairs from $\{1, \ldots, q\}$ defined by: $(r,s) < (i,j)$ if and only if either $r < i$ or ($r = i$ and $s < j$). For each $(i,j) \in \mathcal{G}$, let $M^0_{i,j}$ (respectively, $M^1_{i,j}$) be the matrix whose rows are vectors $I_r + I_s$ with $(r,s) \in \mathcal{G}$ and $(r,s) < (i,j)$ (respectively, $(r,s) \leq (i,j)$). Construct a matrix A by putting $I_i + I_j$ into it if and only if the rank of $M^1_{i,j}$ is greater than that of $M^0_{i,j}$. Simultaneously, construct the vector $(d_1, \cdots, d_m)$.

4. For each i, $1 \leq i \leq q$, let $A_{\leq i}$ be the matrix constructed from A by eliminating all the columns after the i-th one. Construct matrix A_1 from A by eliminating the i-th column if and only if $A_{\leq i}$ and $A_{\leq i-1}$ have the same rank. Put the remaining columns in A_2.

Mulmuley [Mul87] constructed an NC^2 algorithm to compute the rank of a matrix. All calls to the reduction f are made in parallel in step 2, and can be executed in NC^2, since f is logspace computable. Therefore, steps 2-4 of the algorithm can be implemented in NC^2, and the overall complexity of the resulting algorithm is RNC^2.

2.3 Derandomizing the construction

The above construction can be derandomized by using the following small bias sample space of [AGHP92] instead of a random sample of n-bit vectors. Consider the Galois field $\mathrm{GF}(2^h)$ of size 2^h. Since $\mathrm{GF}(2^h)$ is isomorphic to the set of residue classes of $\mathrm{GF}(2)[X]$ modulo an irreducible polynomial of degree h over $\mathrm{GF}(2)$, each element of $\mathrm{GF}(2^h)$ can be represented as an h-dimensional $\{0,1\}$ vector. For $u, v \in \mathrm{GF}(2^h)$, let $\langle u, v\rangle$ denote the inner product $\sum_{i=1}^{h} u_i v_i$ modulo 2 where $u = (u_1, \ldots, u_h)$ and $v = (v_1, \ldots, v_h)$, and define $I(u, v)$ to be the vector

$$(\langle 1, v\rangle, \langle u^1, v\rangle, \langle u^2, v\rangle, \ldots, \langle u^{n-1}, v\rangle).$$

Cai and Sivakumar showed that using $\{I(u, v) \mid u, v \in \mathrm{GF}(2^h)\}$ as the set $\mathcal{F}$ in step 1 of the above algorithm always produces a system of rank $n - O(\log n)$, provided $h \in \Omega(\log n)$. A Galois field of size 2^h with $h \in O(\log n)$ can be constructed in logspace: one only has to try all polynomials over $\mathrm{GF}(2)$ of degree h and select the first irreducible one by trial division. Each element in $\mathrm{GF}(2^h)$ is of length h, and $I(u, v)$ is logspace computable given u and v. Thus, the resulting algorithm has complexity NC^2.

2.4 The final step: NC^1-simulation

The major obstacle to improving the $\mathrm{P} = \mathrm{NC}^2$ to the ultimate $\mathrm{P} = \mathrm{L}$ collapse is the fact that no algorithms better than NC^2 are known for computing the rank or for solving systems of linear equations. In order to overcome the difficulty, one can try to find a method to generate some special form of systems of linear equations for which both computing the rank and solving the system can be done in logspace. Vandermonde systems over $\mathrm{GF}(2^h)$ with $h \in O(\log n)$ have this property.

Define GFCVP to be the set of all quintuples $\langle C, x, 1^h, u, v\rangle$ such that h is of the form $2 \cdot 3^l$, $u, v \in \mathrm{GF}(2^h)$, and $\sum_{i=1}^{n} u^{i-1} g_i = v$, where $g_1, \cdots, g_n$ denote the values of the gates of C on input x, and the arithmetic is over $\mathrm{GF}(2^h)$. We require h to be of the form $2 \cdot 3^l$, because an easy explicit expression for an irreducible polynomial of degree $2 \cdot 3^l$ over $\mathrm{GF}(2)$ is known (namely, $X^{2\cdot 3^l} + X^{3^l} + 1$) [vL91], which allows us to perform all arithmetic involved efficiently. If we manage to find in logspace n pairs $(u_1, v_1), (u_2, v_2), \ldots, (u_n, v_n) \in \mathrm{GF}(2^h)^2$ for some $h \in O(\log n)$ such that $\langle C, x, 1^h, u_i, v_i\rangle \in \mathrm{GFCVP}$, then we are done: We just have to solve the Vandermonde system

$$\begin{pmatrix} 1 & u_1 & \cdots & u_1^{n-1} \\ \vdots & \vdots & & \vdots \\ 1 & u_n & \cdots & u_n^{n-1} \end{pmatrix} \gamma = \begin{pmatrix} v_1 \\ \vdots \\ v_n \end{pmatrix} \tag{3}$$

to recover the gate values $\gamma = (g_1, \cdots, g_n)$. It is known that solving full rank Vandermonde systems can be done in NC^1 using arithmetic circuits over any field F [Ebe89]. In general, for a finite field of characteristic c and size $s = c^h$, an $n \times n$ full rank Vandermonde system over $\mathrm{GF}(c^h)$ is solvable using boolean circuits of depth $O(\log c \log n)$

which can be generated in space $O(h \log c + \log n)$. In our case, $c = 2$ and $h \in O(\log n)$, so the system is solvable using logspace-uniform NC^1 boolean circuits.

Now let us turn to the problem of constructing such a system for fixed C and x. Consider the set $\mathcal{F} = \{\langle C, x, 1^h, u, v\rangle \mid u, v \in \mathrm{GF}(2^h)\}$. Note that for each $u \in \mathrm{GF}(2^h)$, there is a unique $v \in \mathrm{GF}(2^h)$ such that $\langle C, x, 1^h, u, v\rangle \in \mathrm{GFCVP}$. Since GFCVP is in P, by hypothesis there is a $\leq_m^{log}$-reduction f from it to a sparse set S.

Observation 3 *There exists a $y \in S$ such that for at least $2^h/p(n)$ many $z \in \mathcal{F}$, $f(z) = y$.*

In other words, there is a popular element $y \in S$ to which many elements of $\mathcal{F} \cap \mathrm{GFCVP}$ are mapped by f. Based on this observation, we can obtain the gate values as follows:

1. Set h to the smallest integer of the form $2 \cdot 3^l$ such that $2^h/p(n) \geq n$. Note that $h \in O(\log n)$.
2. For each $(s,t) \in \mathrm{GF}(2^h)^2$, in parallel do the following:
   (a) Select the first n pairs $(u_1, v_1), (u_2, v_2), \ldots, (u_n, v_n) \in \mathrm{GF}(2^h)^2$ such that all u_i's are different, and $f(\langle C, x, 1^h, u, v\rangle) = f(\langle C, x, 1^h, s, t\rangle)$. If there are no n such pairs, then give up for this (s,t).
   (b) Solve the Vandermonde system (3) for γ.
   (c) Run the consistency test on the solution. If the test is passed, then output γ.

By the above observation, there is at least one (s,t) from which we obtain the correct gate values. Also, our consistency test is only passed by the correct gate values. Thus, the algorithm correctly computes γ. The complexity of the algorithm is logspace-uniform NC^1: First, since $h \in O(\log n)$, we can implement the looping in step 2 in NC^1. Second, the test in step 2a as well as the setup of the Vandermonde system can be done in logspace-uniform NC^1 by use of the reduction f and by counting. Third, as mentioned earlier, solving the Vandermonde systems obtained and testing consistency are doable in logspace-uniform NC^1. Since f is logspace computable, the overall complexity of the algorithm is logspace. Thus, we have the ultimate collapse.

Theorem 4 *If* P *has a sparse $\leq_m^{log}$-hard set, then* $\mathrm{P} \subseteq \mathrm{L}$.

Since the queries to f are made in parallel, in the case that f is computable in NC^1, we have a stronger collapse.

Corollary 5 *If* P *has a sparse hard set under* NC^1*-computable many-one reductions, then* $\mathrm{P} \subseteq \mathrm{NC}^1$.

3 SPARSE BTT-HARD SETS FOR P

Now, we will extend Theorem 4 to $\leq_{btt}^{log}$-reductions:

Theorem 6 *If* P *has a sparse* $\leq_{btt}^{log}$*-hard set, then* $\mathrm{P} \subseteq \mathrm{L}$.

The outline of the proof is the same as for the many-one case: Using the $\leq_{btt}^{log}$-reduction of GFCVP to a sparse set S, for a given circuit C with n gates and a given input x for C, we will construct in logspace a Vandermonde system in the gate values γ of the form (3) over the field $\mathrm{GF}(2^h)$ where $h \in O(\log n)$. Since we can solve such a system in logspace, we will obtain a logspace algorithm for CVP. Once again, the algorithm will actually be NC^1 modulo the complexity of the reduction, and the reduction will only be used to answer queries that are made in parallel. Hence, we will also obtain:

Corollary 7 *If* P *has a sparse hard set under* NC^1*-computable bounded truth-table reductions, then* $\mathrm{P} \subseteq \mathrm{NC}^1$.

3.1 The main idea

Suppose the reduction from GFCVP to S makes k queries. In the many-one case, we used Observation 3 as the basis for constructing the Vandermonde system. That observation does not hold in the $\leq_{btt}^{log}$ setting: We cannot guarantee the existence of a single popular element (in S or not), let alone of a set of k jointly popular elements.

What can we do in that case, i.e., when no single element of S is queried by the reduction for many inputs from $\mathcal{F}$ that belong to GFCVP? Provided $\mathcal{F} \cap \mathrm{GFCVP}$ is sufficiently large, there must be many inputs from $\mathcal{F} \cap \mathrm{GFCVP}$ that have *all* their queries outside of S, because of the polynomial bound $p(n)$ on the number of elements in the sparse set S that can be queried. Van Melkebeek observed the following crucial point:

Observation 8 *If we order the set* $Q(\mathcal{F})$ *of all queries the reduction makes on inputs from* $\mathcal{F}$ *(say lexicographically), then* S *divides* $Q(\mathcal{F})$ *into a collection of at most* $p(n)+1$ *subintervals of* $Q(\mathcal{F})$.

The relevance of this observation is that it allows us to generate in logspace a subset of $\mathcal{F} \cap \mathrm{GFCVP}$ that contains a fraction of at least $\binom{p(n)+1}{k}^{-1}$ of all the inputs from $\mathcal{F} \cap \mathrm{GFCVP}$ that have all their queries outside of S.

Indeed, every input from $\mathcal{F} \cap \mathrm{GFCVP}$ all of whose queries are outside of S, is mapped by the reduction to a collection of at most k of the $p(n)+1$ (or less) subintervals that S induces in $Q(\mathcal{F})$. So, there is a collection of k of these subintervals such that their union contains all the queries of a fraction of at least $\binom{p(n)+1}{k}^{-1}$ of these inputs. Given the endpoints of the k intervals, we can list these inputs in logspace by cycling through the elements of $\mathcal{F}$ and checking for each of them whether all their queries are in the union of the k intervals and whether the reduction accepts the input assuming all of the queries are outside of S. The inverse polynomial fraction we obtain of the inputs all of whose queries are outside of S, will be large enough for our purposes, as we will see in a moment.

Moreover, we can generate all possible k-subsets of intervals of $Q(\mathcal{F})$ in logspace: We just have to cycle through all possibilities for the endpoints of the k intervals. We can do that in logspace, because we can generate all elements of $\mathcal{F}$ and apply the logspace query generator of the reduction to them.

So, this takes care of the case where there are no popular queries. What about the general case, where there may be popular queries in S? Let us consider a (possibly empty) maximal set Q of jointly popular queries: For many inputs from $\mathcal{F} \cap \mathrm{GFCVP}$, the reduction queries all the elements of Q, and the set Q is maximal with this respect. Hence, we can restrict ourselves to those inputs for which the reduction queries all of Q without loosing too many elements of $\mathcal{F} \cap \mathrm{GFCVP}$. Once we have done that, the maximality of Q implies that the reduction induced on these inputs is a $(k-|Q|)$-truth-table reduction to S without popular queries in S. So, then we can apply the above interval technique. Since a set of jointly popular queries is a subset of the queries the reduction makes on some input of $\mathcal{F}$, we can generate all possibilities for Q by cycling through $\mathcal{F}$ and for each element of $\mathcal{F}$ through all subsets of the queries the reduction makes on that input. We can do that in logspace.

To recapitulate, for every element of $\mathcal{F}$ we will run through all possible subsets Q of queries the reduction makes on that input, and for each Q through all possible collections of $k - |Q|$ subintervals of $Q(\mathcal{F})$. For a fixed Q and collection of intervals, we output those elements of $\mathcal{F}$ for which the reduction queries all of Q and has its remaining queries in the union of the intervals of the collection, and which the reduction accepts assuming the queries in Q belong to S and the remaining ones are outside of

S. It follows from the above discussion that we can implement all of this in logspace, and that we end up with a collection of subsets of $\mathcal{F}$, at least one of which is a subset of GFCVP (because the assumptions we make regarding membership of the reduction's queries to S are correct) which contains a large fraction of $\mathcal{F} \cap \mathrm{GFCVP}$. The remaining piece of the puzzle is checking how large we can make this fraction, and in particular whether we can make the absolute size of the resulting subset of $\mathcal{F} \cap \mathrm{GFCVP}$ to be at least n. Once we manage to do that, we are done, because we can generate in logspace a collection of Vandermonde systems over $\mathrm{GF}(2^h)$ with $h \in O(\log n)$ at least one of which is a full rank $n \times n$ system of the form (3) that correctly describes the gate values γ.

We keep the actual counting for the next section. There we will show that we can obtain a fraction of at least $(e \cdot (p(n) + 1)^k)^{-1}$ of $\mathcal{F} \cap \mathrm{GFCVP}$, where e denotes the base of the natural logarithm. So, we can construct a subset of $\mathcal{F} \cap \mathrm{GFCVP}$ of size at least

$$\frac{|\mathcal{F} \cap \mathrm{GFCVP}|}{e \cdot (p(n) + 1)^k}, \tag{4}$$

and the question is whether we can make this at least n without pushing $h(n)$ above logarithmic. Note that $|\mathcal{F} \cap \mathrm{GFCVP}| = 2^h$. The polynomial p implicitly also depends on h, because it bounds the number of elements of the sparse set S that can be queried by the reduction from GFCVP to S on inputs of $\mathcal{F}$, and these inputs have length $O(n^2 + h(n))$. However, we can fix a polynomial p that is an asymptotic bound as long as $h(n) \in O(\log n)$, and only the value of n starting from which $p(n)$ is an actual bound depends on h. It is then clear that a sufficiently large $h \in O(\log n)$ will (asymptotically) make (4) grow above n, since the denominator of (4) is a fixed polynomial in n.

3.2 The counting argument

We now quantify the argument of the previous section: We will show that there is a subset Q of S and a collection of subintervals $I_1, \ldots, I_{k-|Q|}$ of $Q(\mathcal{F})$ disjoint from S such that the associated subset

$$G = \{z \in \mathcal{F} \mid Q \subseteq Q(z) \text{ and } Q(z) \setminus Q \subseteq \bigcup_{i=1}^{k-|Q|} I_i \text{ and the reduction accepts } z \text{ on } Q\}$$

has size at least d, where $Q(z)$ denotes the set of queries the reduction from GFCVP to S makes on input z, d is the quantity (4), and "the reduction accepts z on Q" means that it accepts z when using Q instead of S to answer membership queries about S.

Given any popularity criterion $q : \{0,\ldots,k\} \to [0,\infty)$, it is obvious that there is a maximal set Q of jointly popular queries which belong to S, i.e., a set $Q \subseteq S$ such that:

$$|\{z \in \mathcal{F} \cap \mathrm{GFVCP} \mid Q \subseteq Q(x)\}| \geq q(|Q|) \tag{5}$$

$$\forall\, w \in S \setminus Q : |\{z \in \mathcal{F} \cap \mathrm{GFVCP} \mid Q \cup \{w\} \subseteq Q(z)\}| \leq q(|Q|+1), \tag{6}$$

provided that $q(0) \leq |\mathcal{F} \cap \mathrm{GFVCP}|$ and we set $q(k+1) = 0$. We will determine q later on, once we know all properties we need of it. We allow q to have real non-integral values because d is non-integral. Note that the range of $|Q|$ is $\{0,\ldots,k\}$.

Consider the set of inputs $G' = \{z \in \mathcal{F} \cap \mathrm{GFVCP} \mid Q(z) \cap S = Q\}$.

Claim 1 $|G'| \geq q(|Q|) - p(n) \cdot q(|Q|+1)$.

Indeed, the set G' contains all elements of the set on the left-hand side of (5), except for those that have at least one query in $S \setminus Q$. Because of (6) the number of exceptions is bounded by $p(n) \cdot q(|Q|+1)$.

Claim 2 *There are intervals $I_1,\ldots,I_{k-|Q|}$ of $Q(\mathcal{F}) \setminus S$ such that:*

$$|\{z \in G' \mid Q(z) \setminus Q \subseteq \bigcup_{i=1}^{k-|Q|} I_i\}| \geq \frac{|G'|}{\max(1, \binom{p(n)+1}{k-|Q|})}.$$

This is because S partitions $Q(\mathcal{F}) \setminus S$ in at most $p(n)+1$ intervals, and for each $z \in G'$, $Q(z) \setminus Q$ is in the union of at most $k - |Q|$ of these intervals.

The combination of Claims 1 and 2 yields that the associated set G satisfies

$$|G| \geq \frac{q(|Q|) - p(n) \cdot q(|Q|+1)}{\max(1, \binom{p(n)+1}{k-|Q|})},$$

which is at least d, provided that for $i \in \{0,\ldots,k\}$

$$\frac{q(i) - p(n) \cdot q(i+1)}{\max(1, \binom{p(n))+1}{k-i})} \geq d.$$

It is straightforward to check that the function

$$q(i) = d \cdot \sum_{j=0}^{k-i} (p(n)+1)^{k-i-j} \binom{p(n)+1}{j}$$

satisfies these conditions. The upper bound for $q(0)$ is also met, since $q(0) \leq d \cdot (p(n)+1)^k \cdot \sum_{j=0}^{k} \frac{1}{j!} \leq |\mathcal{F} \cap \mathrm{GFVCP}|$. The existence of q concludes the formal proof of counting argument.

3.3 Extension to other classes than P

The ideas of the foregoing sections can be used to collapse a complexity class $\mathcal{C}_1$ other than P to a class $\mathcal{C}_2$ that contains NC^1, assuming the existence of a sparse hard set for $\mathcal{C}_1$ under bounded truth-table reductions computable with the power of $\mathcal{C}_2$. In order for the technique to be applicable, sets in $\mathcal{C}_1$ must have unique membership proofs that can be constructed in $\mathcal{C}_1$ and checked in $\mathcal{C}_2$. We can encode the membership proofs as solutions of Vandermonde systems and use the procedure based on Observation 8 to recover them in a space efficient way using a space efficient bounded truth-table reduction of the encoding auxiliary set (GFCVP in case of collapsing P to L) to a sparse set.

This approach works for NL and L. In the case of NL, we can construct a logspace algorithm for the s-t connectivity problem for DAG's, which is complete for NL under logspace many-one reductions. The auxiliary set used here contains all tuples $\langle G, t, u, u^2, u^3, \ldots, u^{n-1}, v\rangle$ such that G is a DAG with n vertices, t is a vertex of G, u and v are elements of $\mathrm{GF}(2^h)$ for some h of the form $2 \cdot 3^l$, and $\sum_{i=1}^{n} u^{i-1} g_i = v$, where g_i is a boolean indicating whether there is a path from the i-th node of G to t. The values $g_1, \ldots, g_n$ constitute a logspace verifiable proof of the existence of an s-t path in the DAG G for any vertex s of G. Note that we include all the powers of u needed to check the equality $\sum_{i=1}^{n} u^{i-1} g_i = v$. This is because we can compute the product of two elements of $\mathrm{GF}(2^h)$ in space $O(\log h)$, but do not know how to compute the n-th power of an element of $\mathrm{GF}(2^h)$ in space $O(\log h + \log n)$. Together with the fact that NL is closed under complement, which allows us to compute the g_i's in NL, this trick suffices to keep the auxiliary set in NL. This way, we obtain

Theorem 9 *If* NL *has a sparse* $\leq_{btt}^{log}$*-hard set, then* $\mathrm{NL} \subseteq \mathrm{L}$.

As in the case of P, we can also prove:

Corollary 10 *If* NL *has a sparse hard set under* NC^1*-computable bounded truth-table reductions, then* $\mathrm{NL} \subseteq \mathrm{NC}^1$.

The s-t connectivity problem for undirected acyclic graphs is complete for L under NC^1-computable many-one reductions, and using the same techniques as above we can devise an NC^1-algorithm for it assuming the existence of a sparse hard set for L under NC^1-computable bounded truth-table reductions. The auxiliary set in L considered here consists of all tuples $\langle G, s, t, u, u^2, u^3, \ldots, u^{n-1}, v\rangle$ such that G is an acyclic graph with n edges, s and t are vertices of G that belong to the same component, u and

v are elements of $\mathrm{GF}(2^h)$ for some h of the form $2 \cdot 3^l$, and $\sum_{i=1}^{n} u^{i-1} g_i = v$, where g_i is a boolean indicating whether the i-th edge of G is on the unique path connecting s and t. The values $g_1, \ldots, g_n$ constitute an NC^1-verifiable proof of the existence of an s-t path in the acyclic graph G. Thus, we get:

Theorem 11 *If* L *has a sparse hard set under* NC^1*-computable bounded truth-table reductions, then* $\mathrm{L} \subseteq \mathrm{NC}^1$.

4 CONCLUSION

Theorem 6 states that P does not have sparse $\leq_{btt}^{log}$-hard sets unless $\mathrm{P} = \mathrm{L}$. Can we show a similar result for $\leq_{tt}^{log}$-reductions? By generalizing the proof described in the previous section, Van Melkebeek showed that for any sublinear function f, the existence of sparse $\leq_{f(n)\text{-}tt}^{log}$-hard sets for P implies that $\mathrm{P} \subseteq \mathrm{Space}[f^2(n^{O(1)}) \log n]$. In case of $\leq_{tt}^{log}$-reductions, the space complexity of the underlying algorithm for CVP blows up to polynomial, so we do not get a collapse at all. Moreover, as in the many-one case, known techniques for classes above P do not appear applicable. One possible approach would be to look at P-complete sparse sets instead of P-hard ones, and then try to extend the result to the P-hard case. But, at this moment, we do not know how to take advantage of the sparse hard set being in P. We hope that the arrival of more sophisticated techniques will shed light on this question in the future.

References

[AGHP92] N. Alon, O. Goldreich, J. Håstad, and R. Peralta. Simple constructions of almost k-wise independent random variables. *Random structures and algorithms*, 3(3):289–304, 1992. Addendum: 4(1):119–120, 1993.

[AHOW92] E. Allender, L. Hemachandra, M. Ogiwara, and O. Watanabe. Relating equivalence and reducibility to sparse sets. *SIAM Journal on Computing*, 21(3):551–539, 1992.

[Ber78] P. Berman. Relationship between density and deterministic complexity of NP-complete languages. In *Proceedings of the 5th Conference on Automata, Languages and Programming*, pages 63–71. Springer-Verlag *Lecture Notes in Computer Science #62*, 1978.

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

[BK88] R. Book and K. Ko. On sets truth-table reducible to sparse sets. *SIAM Journal on Computing*, 17(5):903–919, 1988.

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

[CNS96] J. Cai, A. Naik, and D. Sivakumar. On the existence of hard sparse sets under weak reductions. In *Proceedings of the 13th Symposium on Theoretical Aspects of Computer Science*, pages 307–318, 1996.

[CO96] J. Cai and M. Ogihara. Sparse sets versus complexity classes. In L. Hemaspaandra and A. Selman, editors, *Complexity Theory Retrospective II*. Springer-Verlag, 1996. To appear.

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

[Coo85] S. Cook. A taxonomy of problems with fast parallel algorithms. *Information and Computation*, 64:2–22, 1985.

[CS95a] J. Cai and D. Sivakumar. The resolution of a Hartmanis conjecture. In *Proceedings of the 36th Symposium on Foundations of Computer Science*, pages 362–371. IEEE Computer Society Press, 1995.

[CS95b] J. Cai and D. Sivakumar. The resolution of Hartmanis' conjecture for NL-hard sparse sets. Technical Report 95-40, Department of Computer Science, State University of New York at Buffalo, Buffalo, NY, September 1995.

[Ebe89] W. Eberly. Very fast parallel polynomial arithmetic. *SIAM Journal on Computing*, 18(5):955–976, 1989.

[FFK92] S. Fenner, L. Fortnow, and S. Kurtz. The isomorphism conjecture holds relative to an oracle. In *Proceedings of the 33rd Symposium on Foundations of Computer Science*, pages 30–39. IEEE Computer Society Press, October 1992.

[For79] S. Fortune. A note on sparse complete sets. *SIAM Journal on Computing*, 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.

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

[HOT94] L. Hemaspaandra, M. Ogihara, and S. Toda. Space-efficient recognition of sparse self-reducible languages. *Computational Complexity*, 4:262–296, 1994.

[HOW92] L. Hemachandra, M. Ogiwara, and O. Watanabe. How hard are sparse sets. In *Proceedings of the 7th Conference on Structure in Complexity Theory*, pages 222–238. IEEE Computer Society Press, June 1992.

[KL80] R. Karp and R. Lipton. Some connections between nonuniform and uniform complexity classes. In *Proceedings of the 12th Symposium on Theory of Computing*, pages 302–309. ACM Press, 1980. Final version: Turing machines that take advice, *L'enseignement Mathématique* 28 (1982) 191–209.

[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.

[Ko89] K. Ko. Distinguishing conjunctive and disjunctive reducibilities by sparse sets. *Information and Computation*, 81(1):62–87, 1989.

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

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

[vL91] J. van Lint. *Introduction to coding theory*. Springer-Verlag, 1991.

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

[vM96] D. van Melkebeek. Reducing P to a sparse set using a constant number of queries collapses P to L. In *Proceedings of the 11th Conference on Computational Complexity*, pages 88–96. IEEE Computer Society Press, 1996.

[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 *Proceedings of the 36th Symposium on Foundations of Computer Science*, pages 354–361. IEEE Computer Society Press, 1995.

[OL93] M. Ogiwara and A. Lozano. Sparse hard sets for counting classes. *Theoretical Computer Science*, 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 Journal on Computing*, 20(3):471–483, 1991.

[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).