Sparse Hard Sets for P Yield Space-Efficient Algorithms

Mitsunori Ogihara[1]
Department of Computer Science
University of Rochester
Rochester, NY 14627

Technical Report 569

January, 1995

[1]A.k.a Mitsunori Ogiwara.

Abstract

In 1978, Hartmanis conjectured that there exist no sparse complete sets for P under logspace many-one reductions. In this paper, in support of the conjecture, it is shown that if P has sparse hard sets under logspace many-one reductions, then P $\subseteq$ DSPACE[$\log^2 n$]. The result is derived from a more general statement that if P has $2^{polylog}$ sparse hard sets under poly-logarithmic space-computable many-one reductions, then P $\subseteq$ DSPACE[*polylog*].

1 Introduction

In 1978, Hartmanis conjectured that no P-complete sets under logspace many-one reductions can be polynomially sparse; i.e., for any P-complete set A, $\| \{x \in A \mid |x| \leq n\} \|$ cannot be bounded by any polynomial in n [5]. The conjecture is interesting and fascinating. If the conjecture is true, then L $\neq$ P, because L = P implies any nonempty finite set being P-complete. So, with expectation that L is different from P, one might believe the validity of the conjecture. Nevertheless, such a reasoning would be fallacious, for, proving this conjecture is *at least* as hard as proving L $\neq$ P, and therefore, even though L $\neq$ P, P may have polynomially sparse complete sets. In order to support the conjecture, one would perhaps need to show a result in the other direction; that is, the conjecture does hold unless some 'implausible' collapse of P occurs. What is an implausible collapse of P? As sets in P already have time-efficient recognition algorithms, an implausible collapse should be such that P has 'space-efficient' algorithms, e.g., P is included in DSPACE[$\log^k n$] for some k.

The conjecture is reminiscent of Hartmanis' another conjecture—the Berman-Hartmanis conjecture [2]—that all NP-complete sets under polynomial-time many-one reduction are polynomially isomorphic to each other. If the Berman-Hartmanis conjecture is true, then P $\neq$ NP and polynomially sparse sets cannot be NP-complete. A result to support this conjecture was obtained by Mahaney [9]. He showed that if there is a polynomially sparse hard set for NP, then P = NP; that is, unless NP collapses to the seemingly small class P, NP cannot have sparse complete sets. This has motivated researchers to study a question of whether the same collapse NP = P is indicated from an assumption that NP has polynomially sparse sets under $\leq_r$-reduction for various reducibilities $\leq_r$. The problem is called "the sparse hard set problem for NP," and it has been a central research area for nearly two decades (for a survey, see [12] and [7]). There have been much difficulty, however, in improving Mahaney's result to a more flexible reducibility, even one-truth-table reducibility, until Ogiwara and Watanabe successfully made a remarkable breakthrough to resolve the question up to bounded-truth-table reducibility [10].

In contrast with the sparse hard set problems for NP, not much work has been done on the Hartmanis' conjecture on P—we could call it "the sparse hard set problem for P." The only result that has been proven in this regard is by Hemaspaandra, Ogihara, and Toda [6] that P cannot have *poly-logarithmic* sparse hard sets unless P is included in SC, the class of sets recognized simultaneously in polynomial-time and in poly-logarithmic space. As one can easily see, the result is still too weak because there is a huge gap between polynomially

sparse sets and poly-logarithmic sparse sets.

In this paper, we give the first solution to the sparse hard set problem for P by showing that unless P $\subseteq$ DSPACE[$\log^2 n$], Hartmanis' conjecture holds. The result follows from a theorem of a more general form below, regarding the topologically sorted circuit value problem, which is known to be P-complete [8].

Theorem 1 *Let $d, e \geq 1$ and let S be a set whose density function is bounded by $2^{\mathcal{O}(\log^d n)}$. Suppose* TSCVP*, the topologically sorted circuit value problem, is many-one reducible to S via a function f computable in $\mathcal{O}(\log^e n)$ space. Then* TSCVP *is in* DSPACE[$\log^{de+1} n$].

Then the following corollaries immediately follow from the theorem.

Corollary 2 *Let $d, e \geq 1$. If every set in* P *is many-one reducible to a $2^{\mathcal{O}(\log^d n)}$ sparse set via a function that is $\mathcal{O}(\log^e n)$ space computable, then* P $\subseteq$ DSPACE[$\log^{de+1} n$].

Especially, if P *has polynomially sparse (i.e., $2^{\mathcal{O}(\log n)}$ sparse) hard sets under logspace many-one reductions, then* P $\subseteq$ DSPACE[$\log^2 n$]*, and thus,* P $\neq$ PSPACE*.*

Corollary 3 *If every set in* P *is many-one reducible to a $2^{polylog}$ sparse set via a function that is $\mathcal{O}(polylog)$ space computable, then* P $\subseteq$ DSPACE[*polylog*].

Let us add a few words about the proof. It involves space-efficient recognition of class $\oplus$L [3], a logarithmic space-bounded version of $\oplus$P [11,4]. Assuming that P has such a sparse hard set, we 'reduce' the topologically sorted circuit value problem to a problem in $\oplus$L, which is a natural analog of the circuit value problem. Since $\oplus$L $\subseteq$ DSPACE[$\log^2 n$] (see [3] and [1]), the problem is in DSPACE[$\log^2 n$].

The paper is organized as follows. In section 2, we will define the circuit value problems. In section 3, we prove our main theorem.

2 Circuit Value Problems and Logspace Counting Complexity Classes

A Boolean circuit is a directed acyclic graph C with labeled nodes. Nodes in C with indegree 0 are called *input gates*. Input gates in C are labeled by integers $1, \cdots, n$ so that no two gates have the same label, where n is the number of input gates in C. There is one designated node in C with outdegree 0, which is called the *output gate.* Each gate other than input gates is labeled by a Boolean function chosen from $\{\neg, \wedge, \vee\}$. A gate labeled by

$\neg$ is called a *NOT* gate and has indegree 1. A gate labeled by $\wedge$ (or $\vee$) is called an *AND* gate (an *OR* gate, respectively) and has indegree ≥ 2. A gate g is said to be a *direct input* to a gate g' if there is an arc from g to g' in C.

A Boolean circuit is said to be of *bounded fan-in* if every gate has indegree ≤ 2. It is said to be of *unbounded fan-in* if some gate may have indegree > 2. A Boolean circuit is encoded by its adjacency matrix and the labels of the gates.

Let C be a Boolean circuit of m gates and n inputs and let $x = x_1 \cdots x_n \in \{0,1\}^n$. For each $i, 1 \leq i \leq m$, let g_i denote the i-th gate in C. For $i, 1 \leq i \leq m$, the output of g_i in C on input x, denoted by $C[x,i]$, is determined inductively as follows:

- If g_i is an input gate labeled by j, then $C[x,i] = x_j$.
- If g_i is a NOT gate whose unique direct input is g_j, then $C[x,i] = \neg(C[x,j])$.
- If g_i is an AND gate and its direct inputs are $g_{j_1}, \cdots, g_{j_k}$, then $C[x,i] = C[x,j_1] \wedge \cdots \wedge C[x,j_k]$.
- If g_i is an OR gate and its direct inputs are $g_{j_1}, \cdots, g_{j_k}$, then $C[x,i] = C[x,j_1] \vee \cdots \vee C[x,j_k]$.

The output of C on input x, denoted by $C(x)$, is $C[x,\hat{i}]$, where $\hat{i}$ is the output gate of C.

A circuit C is said to be *topologically sorted* if for every i, j, if g_i is a direct input gate of g_j, then $i < j$. Ladner [8] showed that the circuit value problem, the problem of deciding whether a circuit C outputs 1 on input x, is complete for P under logspace many-one reduction. One can easily observe that the construction can be actually used to show that the topologically sorted version of the problem is complete under logspace many-one reduction. We will use TSCVP to denote the set of strings $C\#x$ such that C is a Boolean circuit of n inputs, $|x| = n$, and $C(x) = 1$.

Parity function, denoted by $\oplus$, is one that maps n to n modulo 2. A parity gate is a gate of unbounded fan-in that, given nonnegative integers $a_1, \cdots, a_n$ as inputs, computes $\oplus(a_1 + \cdots + a_n)$. By convention, we will often use both $\oplus(a_1, \cdots, a_n)$ and $a_1 \oplus \cdots \oplus a_n$ to denote $\oplus(a_1 + \cdots + a_n)$. A parity circuit is an unbounded fan-in arithmetic circuit in which all the gates compute $\oplus$. The *parity circuit problem* is defined as a variation of the circuit value problem, in which it is asked whether a parity circuit outputs 1 on a specified input. We define Parity-CVP to be the set corresponding to the problem. It is the set of all strings of the form $C\#x$ such that C is a parity circuit and, on input x, outputs 1.

A set L is in $\oplus$L [3] if there exists a logarithmic space-bounded nondeterministic Turing machine N such that for every x, $x \in L$ if and only if the number of accepting computation paths of N on x is odd. It is known [3] that $\oplus\mathrm{L} \subseteq \mathrm{DSPACE}[\log^2 n]$.

Proposition 4 Parity-CVP *belongs to* $\oplus$L.

Proof Let C be a parity circuit of m gates $g_1, \cdots, g_m$ and n input gates $g_{\sigma(1)}, \cdots, g_{\sigma(n)}$ and let $x = x_1 \cdots x_n$. Note for any $a, b, c \in \{0, 1\}$, that $\oplus(a, b, c) = \oplus(\oplus(a, b), c) = \oplus(a, \oplus(b, c))$.

For each $i, 1 \leq i \leq m$, let $\mu(i)$ denote the number of paths in C on x from an input gate with value 1 to the gate g_i. We claim for any $i, 1 \leq i \leq m$, that $C[x, i] = \oplus(\mu(i))$. This is proven by induction.

For the base case, suppose that gate g_i is an input gate. Then the gate outputs 1 on input x if and only if the bit of x assigned to the gate is 1. Trivially, there is exactly one path from the gate to itself. So, the claim holds.

For the induction step, let g_i be a gate in C and let $g_{h_1}, \cdots, g_{h_l}$ be an enumeration of all gates that are direct inputs to g_i. Clearly, $\mu(i) = \sum_{j=1}^{l} \mu(h_j)$. Suppose that the claim holds for $h_1, \cdots, h_l$, i.e., $C[x, h_j] = \oplus(\mu(h_j))$ for all $j, 1 \leq j \leq l$. By definition, $C[x, i] = \oplus(C[x, h_1] + \cdots + C[x, h_l])$. So, $C[x, i] = \oplus(\sum_{j=1}^{l} \oplus(\mu(h_j))) = \oplus(\sum_{j=1}^{l} \mu(h_j)) = \oplus(\mu(i))$. Thus, the claim holds for i. Hence the claim holds for any i.

Now let N be a nondeterministic Turing machine that, on input $C\#x$, behaves as follows:

(i) N nondeterministically guesses $l \in \{1 \cdots m\}$, where m is the number of gates in C.

(ii) For $i := 1$ to l, N nondeterministically guesses $j, 1 \leq j \leq m$, and depending on the value of i, does the following:

- **(a)** If $i = 1$, then N tests whether the j-th gates is an input gate assigned value 1 on input x.

  If the test fails, N immediately rejects. Otherwise, after setting r to j, N proceeds to the next i.

- **(b)** If $1 < i < l$, then N tests whether the r-th gate is a direct input to the j-th gate.

  If the test fails, N immediately rejects. Otherwise, after setting r to j, N proceeds to the next i.

- **(c)** If $i = l$, then N tests whether the r-th gate is a direct input to the j-th gate.

  If the test fails, N immediately rejects. Otherwise, N accepts if and only if the j-th gate is the output gate of C.

It is easy to see that N uses $\mathcal{O}(\log m)$ space and that the number of accepting computation paths of N on $C\#x$ is equal to the number of paths in C on x from an input gate with value 1 to the output gates. So, by the previous discussion, N on $C\#x$ has an odd number of accepting computation paths if and only if $C\#x$ is in Parity-CVP. This proves the proposition. □

3 Proof of Theorem 1

Suppose that the hypothesis of the theorem holds. Define a set A as follows:

$$A = \{C\#x\#i_1\#\cdots\#i_p\#b \mid C \text{ is a topologically sorted Boolean circuit with } m \text{ gates and } n \text{ inputs, } x \in \{0,1\}^n,\ 1 \le i_1 < \cdots < i_p \le m,\ b \in \{0,1\}, \text{ and } \oplus(C[x,i_1],\cdots,C[x,i_p]) = b\ \}.$$

Since TSCVP is in P, obviously $A \in$ P. So, by our supposition, A is many-one reducible to a set S whose density is bounded by $2^{\mathcal{O}(\log^d n)}$ via a function f that is computable in $\mathcal{O}(\log^e n)$ space. For each $n \ge 0$, let S_n be the set of all y in S for which there exists some $w, |w| \le 2n$, such that $f(w) = y$. Then there is a constant $c > 0$ depending only on S and f such that for every n, it holds that:

$$\| S_n \| < 2^{c(\log^{de} n)-1}. \tag{1}$$

Let C be a topologically sorted bounded fan-in Boolean circuit with m gates and n input gates and let $x \in \{0,1\}^n$. Suppose we wish to know whether $C\#x$ belongs to TSCVP. Let $g_1,\cdots,g_m$ denote the gates of C and $g_{\sigma(1)},\cdots,g_{\sigma(n)}$ denote the input gates. Let $Z = \{\sigma(1),\cdots,\sigma(n)\}$. Since C is topologically sorted, without loss of generality, we may assume that g_m is the output gate. Otherwise, the gates of larger labels than that of the output gate are not relevant to the output of C at all, and thus, can be eliminated from C. So, in order to test whether $C\#x$ belongs to TSCVP, we have only to compute $C[x,m]$.

For simplicity, let $\ell = |C\#x|$ and $B = \lceil c \log^{de} \ell \rceil$. Clearly, $\ell > m^2 \ge n^2$. Let $\mathcal{T}$ be the set of all nonempty subsets of $\{1,\cdots,m\}$ of size at most B. For each $I \in \mathcal{T}$ and $b \in \{0,1\}$, we will use $\#i_1\#\cdots\#i_p\#b$ to denote the pair, where $i_1,\cdots,i_p$ is an enumeration of all $i \in I$ in increasing order, and let $w(I,b)$ denote $C\#x$ followed by the encoding of (I,b) and $\alpha(I,b)$ denote $f(w(I,b))$. Clearly, for all but finitely many m, it holds that $|w(I,b)| \le 2|C\#x|$ for any $I \in \mathcal{T}$ and $b \in \{0,1\}$. Let $\tilde{S}$ be the set of all $y \in S$ such that $y = \alpha(I,b)$ for some $I \in \mathcal{T}$ and $b \in \{0,1\}$. Then by (1) above, $\tilde{S} \subseteq S_{|C\#x|}$, and thus,

$$\| \tilde{S} \| \le 2^{B-1}. \tag{2}$$

Let $i \in \{1, \cdots, m\} \setminus Z$. We say that i is *good* if there exist $(I, b), (J, b') \in \mathcal{T} \times \{0, 1\}$ such that

$$\alpha(I, b) = \alpha(J, b') \text{ and } i = \max(I \triangle J), \tag{3}$$

where $I \triangle J$ denotes the symmetric difference of I and J. We say that $i \in \{1, \cdots, m\} \setminus Z$ is *bad* if i is not good.

We claim that the number of bad i is at most $B - 1$. Assume that there are more than $B - 1$ bad i and let $h_1, \cdots, h_B$ be an enumeration of B bad i. Let $\mathcal{R}$ be the set of all nonempty subsets of $\{h_1, \cdots, h_B\}$. Note for any $I \in \mathcal{R}$, that exactly one of $\alpha(I, 0)$ or $\alpha(I, 1)$ is in $\tilde{S}$ because exactly one of $w(I, 0)$ or $w(I, 1)$ is in A. So, let b_I be the unique $b \in \{0, 1\}$ such that $\alpha(I, b) \in \tilde{S}$. Note also, for any distinct $I, J \in \mathcal{R}$, that

$$\{\alpha(I, 0), \alpha(I, 1)\} \cap \{\alpha(J, 0), \alpha(J, 1)\} = \emptyset.$$

Otherwise, for such I and J, since the largest element in $I \triangle J$ is some h_k, (3) holds for I, J and some b, b', and thus, h_k is good, a contradiction. Therefore, for any distinct $I, J \in \mathcal{R}$, $\alpha(I, b_I) \neq \alpha(J, b_J)$. So, there are at least $2^B - 1$ many elements in $\tilde{S}$, which contradicts (2).

Now let $H = \{h_1, \cdots, h_q\}$ be the set of all bad i. By the previous discussion, $q \leq B - 1$. For each good i, let $(I(i), b(I), J(i), b'(i))$ be the smallest (I, b, J, b) witnessing that i is good in the lexicographic order that is naturally induced from the encoding of I, b, J, b'. Define a parity circuit D with $m + 1$ gates and $n + q + 1$ input gates as follows:

- The gates of D are those of C plus one new gate g_0.
- The input gates of D are g_0, the input gates of C, and the bad gates; that is, they are $g_0, g_{\sigma(1)}, \cdots, g_{\sigma(n)}, g_{h_1}, \cdots, g_{h_q}$ and labeled in this order. We will consider only input 1 to g_0.
- Each gate g_i in D other than the inputs compute parity, and its direct inputs are
  - all g_j such that $j \neq i$ and $j \in I(i) \triangle J(i)$ if $b(i) = b'(i)$; and
  - g_0 and all g_j such that $j \neq i$ and $j \in I(i) \triangle J(i)$ otherwise.

Note that D is topologically sorted since C is topologically sorted and if i is good then i is the largest in $I(i) \triangle J(i)$.

For each $v \in \{0, 1\}^q$ and $i, 1 \leq i \leq q$, we say that v is *correct* at i if, depending on the type of g_{h_i} in C, the following conditions are satisfied:

(a) If g_{h_i} is a NOT-gate in C whose direct input is g_j, then $v_i = \neg(D[1xv, j])$.

(b) If g_{h_i} is an AND-gate in C whose direct inputs are g_j and g_k, then then $v_i = D[1xv, j] \wedge D[1xv, k]$.

(c) If g_{h_i} is an OR-gate in C whose direct inputs are g_j and g_k, then then $v_i = D[1xv, j] \vee D[1xv, k]$.

We say that v is *valid* if v is correct at every $i, 1 \leq i \leq q$.

We claim that there is a unique valid v and that the unique valid v, denoted by $\tilde{v}$, satisfies $D[1x\tilde{v}, i] = C[x, i]$ for all $i, 1 \leq i \leq m$. To see this, let $v \in \{0,1\}^q$ and suppose there is some $i, 1 \leq i \leq m$, such that $D[1xv, i] \neq C[x, i]$. We can choose the smallest such i. So, let i be the smallest. Since both C and D are topologically sorted, for any $j \geq 1$ such that g_j is a direct input gate of g_i, $j < i$, so, by the minimality of i, it holds that $D[1xv, j] = C[x, j]$.

Suppose that i is good. Then by definition, $\alpha(w(I(i), b(i)) = \alpha(w(J(i), b'(i))$ Thus,

$$b(i) \oplus \bigoplus_{j \in I(i)} C[x, j] = b'(i) \oplus \bigoplus_{j \in J(i)} C[x, j].$$

This implies

$$C[x, i] = C[x, j_1] \oplus \cdots \oplus C[x, j_k] \oplus b(i) \oplus b'(i),$$

where $j_1, \cdots, j_k$ is an enumeration of all j such that $j \neq i$ and $j \in I(i) \triangle J(i)$. Since $j_1, \cdots, j_k < i$, by the minimality of i, we have

$$C[x, i] = D[1xv, j_1] \oplus \cdots \oplus D[1xv, j_k] \oplus b(i) \oplus b'(i).$$

Suppose $b(i) = b'(i)$. Then, by definition, in D, $g_{j_1}, \cdots, g_{j_k}$ are the direct inputs of g_i. So,

$$D[1xv, i] = D[1xv, j_1] \oplus \cdots \oplus D[1xv, j_k],$$

which implies $C[x, i] = D[1xv, i]$. On the other hand, suppose that $b(i) \neq b'(i)$. Then, by definition, in D, $g_0, g_{j_1}, \cdots, g_{j_k}$ are the direct inputs of g_i. Noting that $D[1xv, 0] = b(i) \oplus b'(i) = 1$, we have

$$D[1xv, i] = D[1xv, 0] \oplus D[1xv, j_1] \oplus \cdots \oplus D[1xv, j_k],$$

which implies $C[x, i] = D[1xv, i]$. This contradicts our assumption that $C[x, i] \neq D[1xv, i]$. Therefore, i is bad, so $i = h_t$ for some $t, 1 \leq t \leq q$. This implies $D[1xv, i] = v_t$. Since C is topologically sorted, by the minimality of i, for any j such that g_j is a direct input to

g_i in C, it holds that $i < j$, and thus, $C[x,i] = D[1xv,i]$. Suppose g_i is a NOT gate in C with direct input g_j. Then $C[x,i] = \neg(C[x,j]) = \neg(D[1xv,j])$. Since $C[x,i] \neq D[1xv,i]$ by our assumption, and $D[x1v,i] = v_t$ as $i = h_t$, we have $v_t \neq \neg(D[1xv,j])$, and thus, v is not correct at t. This implies that $v_t \neq C[x,i]$ and that v is incorrect at t. Similarly, if g_i is an AND gate in C with direct inputs g_j and g_k, then $C[x,i] = C[x,j] \wedge C[x,k] = D[1xv,j] \wedge D[1xv,k]$ and, by our assumption, $C[x,i] \neq D[1xv,i]$. So, the condition (b) is not satisfied for t, and thus, $v_t \neq C[x,i]$, which implies that v is not correct at t. By repeating the same argument again with $\vee$ in place of $\wedge$, we obtain that $v_t \neq C[x,i]$ and that v is not correct at t in the case that g_i is an OR gate. Therefore, v is not correct at t.

From the above discussion, we obtain that if $C[x,i] \neq D[1xv,i]$ for some i, then there exists some good $i = h_t$ such that

- for any $j < i$, $C[x,j] = D[1xv,j]$;
- $v_t \neq C[x,i]$;
- v is not correct at t, and thus, v is not valid; and
- for every $t' < t$, v is correct at t'.

On the other hand, suppose $C[x,i] = D[1xv,i]$ for every i. Then for every $t, 1 \leq t \leq q$, $v_t = D[1xv,h_t] = C[x,h_t]$. So, $v = C[x,h_1] \cdots C[x,h_q]$. Also, for every t, depending on the type of g_{h_t} in C, the conditions (a), (b), and (c) are satisfied. So, v is correct at every t, and thus, valid. Hence, $\tilde{v} = C[x,h_1] \cdots C[x,h_q]$ is a unique valid v and $C[x,i] = D[1x\tilde{v},i]$ for every i. This establishes the claim.

Now, since $C[x,i] = D[1x\tilde{v},i]$ for every i, we have $C(x) = C[x,m] = D[1x\tilde{v},m]$. This suggests the following algorithm to test the membership of $C\#x$ in TSCVP.

Step 1: For each $i, 1 \leq i \leq m$, test whether i is good, and compute H, the set of all bad i.

Step 2: For each $v \in \{0,1\}^q$, test whether v is valid and if so, compute $D[1xv,m]$.

We explain how space-efficient this algorithm can be. Recall for any $I \in \mathcal{T}$ and $b \in \{0,1\}$, that (I,b) is encoded into a string $\#i_1\# \cdots \#i_p\#b$ so that appending the encoding to $C\#x$ yields $w(I,b)$. Since each $i_j \in I$ can be encoded into a string of length at most $\lceil \log(m+1) \rceil$, the length of the encoding of (I,b) will be $\mathcal{O}(B \log m)$.

Let M be a deterministic Turing machine that computes the reduction f in $\mathcal{O}(\log^e n)$ space. Given (I,b) and (J,b'), in order to test whether $\alpha(I,b) = \alpha(J,b')$, it suffices to

simulate M on $w(I,b)$ and M on $w(J,b')$ simultaneously and compare $f(I,b)$ and $f(J,b')$ bit-by-bit. Since M's output tape is certainly write-only, the comparison requires to store only the most recent output bit from each. More precisely, M' on $w(I,b)$ and M' on $w(J,b')$ are simulated alternatively step-by-step. If one side of the simulations outputs a new bit of f, then the simulation on this side is suspended until the other side produces a new bit of f or halts without outputting a new bit. If both sides produce new bits, then the bits are compared are if they are different, it is found that the values of f are different. If only one side produces a new bit, then the two values of f do not have the same length, and thus, are different. The simulations are terminated when both sides halt without producing any new bits. Since the bits they have produced so far are the same, it is found that they have the same value. The amount of space expended by the simulations is $\mathcal{O}(\log^e \ell)$ if we ignore the amount of space required to store I, b, J, b' since simply appending (I,b) and (J,b') to the $C\#x$ yield $w(I,b)$ and $w(J,b')$. So, even if take into account the amount of space to store I, b, J, b', the comparison can be done in $\mathcal{O}(\log^e \ell)$ space.

In order to test whether i is good, it suffices to test whether there is some (I,b,J,b') witnessing that i is good by cycling through all possible (I,b,J,b'). From the above discussion, we can argue that the amount of space expended is $\mathcal{O}(B \log m + \log^e \ell) = \mathcal{O}(\log^{de+1} \ell)$. Also, for a good i, finding $(I(i), b(i), J(i), b'(i))$ requires the same order of space, because the quadruple is lexicographically the first one witnessing that i is good. Since there are at most $B-1$ bad i, the amount of space required to store H, the set of all bad i, is $\mathcal{O}(B \log m) = \mathcal{O}(\log^{de+1} \ell)$ and the amount of space required to store $v \in \{0,1\}^q$ is B.

Given H and v, producing the description of D requires $\mathcal{O}(B \log m) = \mathcal{O}(\log^{de+1} \ell)$ space. To see this, note that there is an arc from g_i to g_j if and only if j is good (i.e., not in H) and either $i \in I(j) \triangle J(j)$ or $(i = 0$ and $b(j) \neq b'(j))$. Since the ordering of the gates in D is the same as that in C except for that g_0 is added, in order to compute the (i,j)-th entry of the adjacency matrix of D, it suffices to compute $I(j), J(j), b(j), b'(j)$. This establishes the claim.

Now in order to test whether v is correct at i, as h_i can be determined from the description of H and the type of the gate g_{h_i} and its direct input(s) can be determined from $C\#x$, it suffices to compute $D[1xv, j]$ for j such that g_j is a direct input to g_{h_i} in C. The computation problem can be reduced to Parity-CVP by simply asking whether D on $1xv$ outputs 1 if gate g_j is assumed to be the output. Let M' be a deterministic Turing machine that decides Parity-CVP in $\mathcal{O}(\log^2 n)$ space. In order to compute $D[1xv, j]$, it suffices to simulate M' by keeping track the position of its input head and when M' needs to read a

k-th bit of its input, then activate the algorithm to produce $D\#1xv$ to compute the k-th bit (by recording only the number of bits produced so far and the current bit). Therefore, $D[1xv, j]$ can be computed in $\mathcal{O}(\log^{de+1} \ell + \log^2 \ell)$ space. Since $d, e \geq 1$, it can be computed in $\mathcal{O}(\log^{de+1} \ell)$ space. Therefore, correctness of v at i, and thus, validity of v can be tested in $\mathcal{O}(\log^{de+1} \ell)$ space. Hence, finding $\tilde{v}$ requires $\mathcal{O}(\log^{de+1} \ell)$ space.

Finally, after finding $\tilde{v}$, $C(x)$ is determined from $C[x, m]$, which is determined from whether D outputs 1 on input $1x\tilde{v}$. Again, this requires $\mathcal{O}(\log^{de+1} \ell)$ space. Therefore, the whole process can be done in $\mathcal{O}(\log^{de+1} \ell)$ space. This proves the theorem.

4 Open Question

We have given a solution to Hartmanis' conjecture on sparse complete sets for P by showing that P cannot not have many-one hard sets of low density via space-efficient reductions unless P $\subseteq$ DSPACE[*polylog*]. One interesting question is whether our proof can be extended to one-truth-table reducibility. But, the question seems subtle. The difficulty is in that, in the one-truth-table reduction case, we cannot argue any longer that at least one of $f(I, b)$ or $f(I, b')$ generates an element belonging to S, and thus, the counting argument for the number of bad i is no longer applicable. Another interesting open question is whether the collapse can be strengthened when we assume completeness of S.

Acknowledgment

The author would like to thank Eric Allender, Lane Hemaspaandra, and Ioan Macarie for valuable comments.

References

[1] C. Àlvarez and B. Jenner. A very hard log-space counting class. *Theoretical Computer Science*, 107:3–30, 1993.

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

[3] G. Buntrock, C. Damm, U. Hertrampf, and C. Meinel. Structure and importance of Logspace-MOD class. *Mathematical Systems Theory*, 25:223–237, 1992.

[4] L. Goldschlager and I. Parberry. On the construction of parallel computers from various bases of boolean functions. *Theoretical Computer Science*, 43:43–58, 1986.

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

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

[7] 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, 1992.

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

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

[10] M. Ogiwara and O. Watanabe. On polynomial time bounded truth-table reducibility of NP sets to sparse sets. *SIAM Journal on Computing*, 20:471–483, 1991.

[11] C. Papadimitriou and S. Zachos. Two remarks on the power of counting. In *Proceedings of the 6th GI Conference on Theoretical Computer Science*, pages 269–276. Springer-Verlag *Lecture Notes in Computer Science #145*, 1983.

[12] 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).