Chicago Journal of Theoretical Computer Science

The MIT Press

Volume 1996, Article 2

27 March 1996

ISSN 1073–0486. MIT Press Journals, 55 Hayward St., Cambridge, MA 02142 USA; (617)253-2889; *journals-orders@mit.edu, journals-info@mit.edu*. Published one article at a time in LaTeX source form on the Internet. Pagination varies from copy to copy. For more information and other articles see:

- *http://www-mitpress.mit.edu/jrnls-catalog/chicago.html*
- *http://www.cs.uchicago.edu/publications/cjtcs/*
- *gopher.mit.edu*
- *gopher.cs.uchicago.edu*
- anonymous *ftp* at *mitpress.mit.edu*
- anonymous *ftp* at *cs.uchicago.edu*

The *Chicago Journal of Theoretical Computer Science* is abstracted or indexed in *Research Alert®, SciSearch®, Current Contents®/Engineering Computing & Technology*, and *CompuMath Citation Index®*.

The *Chicago Journal of Theoretical Computer Science* is a peer-reviewed scholarly journal in theoretical computer science. The journal is committed to providing a forum for significant results on theoretical aspects of all topics in computer science.

Sparse Hard Sets for P Yield Space-Efficient Algorithms

Mitsunori Ogihara

27 March, 1996

Abstract

Abstract-1 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 $P \subseteq DSPACE[\log^2 n]$.

1 Introduction

1-1 In 1978, Hartmanis [Har78] conjectured that no P-complete sets under logspace many-one reductions can be polynomially sparse; i.e., for any set $A \in P$ to which every set in P is logspace many-one reducible, the function that maps n to the number of elements in A of length up to n is not bounded by a polynomial in n. The conjecture is interesting and fascinating. If the conjecture is true, then $L \neq P$, because $L = P$ implies that every nonempty finite set is P-complete. So, since it is widely assumed that L is different from P, one might believe the validity of the conjecture. Such reasoning may well be fallacious: proving this conjecture is *at least* as hard as proving $L \neq P$, and therefore, it may well be the case that even though $L \neq P$, P has polynomially sparse complete sets. In order to support the conjecture, one would perhaps need to show a result in the other direction; that is, if the conjecture *does not* hold, then some "implausible" event occurs. Such an implausible event would be the collapse of P to a (presumably) much smaller class. As sets in P already have time-efficient recognition algorithms, this should mean that P has space-efficient algorithms, e.g., P is included in $DSPACE[\log^k n]$ for some k.

1-2 The conjecture is reminiscent of the celebrated Berman-Hartmanis conjecture [BH77] that all NP-complete sets under polynomial-time many-one reduction are polynomially isomorphic. If the Berman-Hartmanis conjecture is true, then $\mathrm{P} \neq \mathrm{NP}$ and polynomially sparse sets cannot be NP-complete. A result to support this conjecture was obtained by Mahaney [Mah82]. He showed that if there is a polynomially sparse hard set for NP, then $\mathrm{P} = \mathrm{NP}$; that is, unless NP collapses to the seemingly small class P, NP cannot have sparse complete sets.

1-3 In contrast with the sparse hard set problem for NP, not much work has been done on the Hartmanis conjecture for P—we could call it "the sparse hard set problem for P." The only paper I am aware of is by Hemaspaandra, Ogihara, and Toda [HOT94], who prove 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 the poly-logarithmic sparsity is a much more stringent condition than polynomial sparsity.

1-4 In this paper, I give the first solution to the sparse hard set problem for P by showing that unless $\mathrm{P} \subseteq \mathrm{DSPACE}[\log^2 n]$, the Hartmanis conjecture holds for P.

Theorem 1 *There exist no sparse* P*-hard sets under logspace many-one reductions unless* $\mathrm{P} \subseteq \mathrm{DSPACE}[\log^2 n]$.

1-5 Let us say a few words about the proof. Assuming the existence of a sparse P-hard set, we are able to reduce, in a space-efficient manner, any instance of the circuit value problem to Parity-CVP, the circuit value problem of a circuit consisting exclusively of parity gates. However, this restricted circuit value problem is known to be in $\mathrm{DSPACE}[\log^2 n]$.

1-6 Readers familiar with the class $\oplus\mathrm{L}$ [BDHM92], a logarithmic space-bounded version of $\oplus\mathrm{P}$ [PZ83, GP86], will recognize our reduction to map an instance of the circuit value problem to a problem in $\oplus\mathrm{L}$, and recall that $\oplus\mathrm{L} \subseteq \mathrm{DSPACE}[\log^2 n]$ (see [BDHM92] and [ÀJ93]).

1-7 The paper is organized as follows. In Section 2, I define the basic notation and the circuit value problems. In Section 3, I prove the main theorem.

2 Circuit Value Problems

2-1 A Boolean circuit is a directed acyclic graph C with labeled nodes. Nodes in C with indegree 0 are called *input gates*, while the other gates are called *interior gates*. Input gates in C have distinct labels from $\{1, \ldots, n\}$, 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 interior gate 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.

2-2 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, where I always assume that the output gate is the last node and for every i, the i-th input gate is the i-th node.

2-3 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 $g_{j_1}, \ldots, g_{j_k}$, 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 $g_{j_1}, \ldots, g_{j_k}$, 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 $C[x,m]$.

2-4 The circuit value problem (CVP) is the problem of deciding whether a bounded fan-in Boolean circuit C outputs 1 on input x. Ladner [Lad75] showed that CVP is complete for P under logspace many-one reductions. A circuit C is *topologically sorted* if for every i, j, if g_i is a direct input gate of g_j, then $i < j$. One can easily observe that the construction by Ladner can be used to show that the topologically sorted version of the problem, TSCVP, is complete under logspace many-one reductions. I will identify

TSCVP with the set of all strings $C\#x$ such that C is a topologically sorted Boolean circuit of n inputs, $|x| = n$, and $C(x) = 1$.

2-5 The parity function, denoted by $\oplus$, maps a binary string to the parity of the number of 1s in it. I also view $\oplus$ as the function that maps a natural number n to (n modulo 2). A parity (or, exclusive-or) gate is a gate of unbounded fan-in that, given binary bits $a_1, \ldots, a_n$ as inputs, computes $\oplus(a_1 \cdots a_n)$. By convention, I will 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 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. I define Parity-CVP to be the set corresponding to the problem: The set of all $C\#x$ such that C is a parity circuit and, on input x, outputs 1.

2-6 A set L is in $\oplus$L [BDHM92] 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.

Proposition 2 Parity-CVP *is in* $\oplus$L.

Proof of Prop 2-1 **Proof of Proposition 2** Let C be a parity circuit of m gates $g_1, \ldots, g_m$ and n input gates, and let $x = x_1 \cdots x_n$. Note for any $a, b, c \in \{0, 1\}$, that $\oplus(a, b, c) = \oplus(\oplus(a, b), c) = \oplus(a, \oplus(b, c))$.

Proof of Prop 2-2 For each i, $1 \le i \le m$, let $\mu(i)$ denote the number of paths in C on x from some input gate with value 1 to the gate g_i. I claim for any i, $1 \le i \le m$, that $C[x, i] = \oplus(\mu(i))$. This is proven by induction.

Proof of Prop 2-3 For the base case, let g_i be an input gate. Then $C[x, i] = 1$ if and only if $x_i = 1$. Trivially, there is exactly one path from the gate to itself. So, the claim holds.

Proof of Prop 2-4 For the induction step, let g_i be an interior gate and let $g_{h_1}, \ldots, g_{h_l}$ be an enumeration of all direct inputs to g_i. Clearly, $\mu(i) = \sum_{j=1}^{l} \mu(h_j)$. Suppose that the claim holds for $h_1, \ldots, h_l$, i.e., $C[x, h_j] = \oplus(\mu(h_j))$ for all $j, 1 \le j \le l$. By definition, $C[x, i] = \oplus(C[x, h_1] + \cdots + C[x, h_l])$. So,

$$C[x, i] = \bigoplus_{j=1}^{l}(\oplus(\mu(h_j))) = \bigoplus_{j=1}^{l}(\mu(h_j)) = \oplus(\mu(i))$$

Thus, the claim holds for g_i. Hence the claim holds for every gate.

Proof of Prop 2-5 Now, noting that Boolean circuits are acyclic, it is easy to construct a nondeterministic machine witnessing Parity-CVP $\in \oplus$L. Our machine, on input $C\#x$, guesses a sequence $g_{i_1}, \ldots, g_{i_h}$ of at most m gates, and accepts if and only if the sequence is a path from an input gate outputting 1 to the output gate. The verification can be done sequentially, so the machine has only to store two consecutive elements in the sequence. So, it can be logarithmic space-bounded. Clearly, the number of accepting computation paths of the machine on $C\#x$ is equal to the number of paths in C on x from some input gate with value 1 to the output gate. So, the machine witnesses that Parity-CVP $\in \oplus$L. This proves the proposition.

Proof of Proposition 2 □

Proposition 3 ([BDHM92, ÀJ93]) $\oplus\mathrm{L} \subseteq \mathrm{DSPACE}[\log^2 n]$.

Here I provide a sketch of the proof, which is reminiscent of Savitch's theorem [Sav70].

Proof of Proposition 3 Let N be a logarithmic space-bounded nondeterministic machine N witnessing that $L \in \oplus$L, and let x be an input to N. For two IDs I and J of N on x and a natural number t, define $Q(I, J, t)$ to be the parity of the number of computation paths of N on x from I to J of length at most 2^t. Define $Q(I, I, t) = 1$ for every I and t. Let $m = O(\log|x|)$ be a natural number such that N on x runs for at most 2^m steps, and let I_{ini} be the unique start ID of N on x. We may assume that there is a unique accepting ID of N on x. Let I_{acc} denote this ID. Clearly, $x \in L$ if and only if $Q(I_{ini}, I_{acc}, m) = 1$. Note for every I, J, and $t > 0$, that

$$Q(I, J, t) = \bigoplus_K (Q(I, K, t-1)Q(K, J, t-1))$$

where K ranges over all IDs of N on x. This suggests the following recursive procedure to evaluate $Q(I, J, t)$:

- If $I = J$, then return 1.
- If $I \neq J$ and $t = 0$, then compute and return $Q(I, J, 0)$ by simulating one move of N on x at ID I.
- If $I \neq J$ and $t > 0$, then set c to 0 and for each K, set c to $(c + Q(I, K, t-1)Q(K, J, t-1))$ modulo 2.

If we run this procedure to evaluate $Q(I_{ini}, I_{acc}, m)$, then the recursion depth is $m = O(\log|x|)$. Since each ID is encoded as a string of length $O(\log|x|)$, the evaluation requires $O(\log^2|x|)$ space, and thus, $L \in \mathrm{DSPACE}[\log^2 n]$.

Proof of Proposition 3 □

2-7 From the above two propositions, I immediately obtain the following:

Proposition 4 Parity-CVP $\in \mathrm{DSPACE}[\log^2 n]$.

3 Proof of Theorem 1

3-1 I repeat the statement of the theorem.

Theorem 1 *There exist no sparse* P*-hard sets under logspace many-one reductions unless* $\mathrm{P} \subseteq \mathrm{DSPACE}[\log^2 n]$.

Proof of Theorem 1-1 **Proof of Theorem 1** Suppose that there exists a sparse P-hard set under logspace many-one reductions. Then I show that $\mathrm{P} \subseteq \mathrm{DSPACE}[\log^2 n]$, in particular, TSCVP is in $\mathrm{DSPACE}[\log^2 n]$.

Proof of Theorem 1-2 I will make use of the following set A: A is the set of all strings of the form $C\#x\#I\#b$ such that:

- $C\#x$ is an instance of TSCVP, i.e., C is a topologically sorted Boolean circuit with m gates and n inputs and $x \in \{0,1\}^n$,
- I is a nonempty subset of $\{1, \ldots, m\}$ encoded as the enumeration of its elements in increasing order,
- $b \in \{0, 1\}$, and
- $\bigoplus_{i \in I} C[x, i] = b$.

Clearly, $A \in \mathrm{P}$. So, by our supposition, A is logspace many-one reducible to a sparse set S via some function f. Note that for a sufficiently large m and every legitimate $C\#x\#I\#b$, it holds that $|C\#x\#I\#b| \leq 2|C\#x|$. Since S is sparse, this implies that for every $C\#x$, the number of $y \in S$ such that $y = f(C\#x\#I\#b)$ for some I and b is bounded by $2^{d \log|C\#x|}$ for some constant d.

Proof of Theorem 1-3 Let $C\#x$ be fixed, whose membership in TSCVP we are testing. Let $g_1, \ldots, g_m$ be the gates of C, where $g_1, \ldots, g_n$ are the input gates and g_m is the output gate. Let $l = |C\#x|$ and $e = \lceil d \log l \rceil$. As we have already fixed C and x, I will simply use $I\#b$ to denote $C\#x\#I\#b$ by dropping C and x. By the above observation, the number of $y \in S$ such that $y = f(I\#b)$ for some $I \in \{1, \dots, m\}$ and $b \in \{0,1\}$ is less than 2^e.

Proof of Theorem 1-4 Now I introduce the notion of *good* gates and *bad* gates. Let $\mathcal{T}$ be the set of all nonempty subsets of $\{1, \dots, m\}$ of size at most e. Let $i \in \{n+1, \dots, m\}$. I say that g_i is good if there exist distinct $I, J \in \mathcal{T}$ and $b, c \in \{0,1\}$ such that

$$f(I\#b) = f(J\#c) \text{ and } i = \max(I \bigtriangleup J)$$

where $I \bigtriangleup J$ denotes the symmetric difference of I and J. Otherwise, g_i is called bad. Intuitively, an interior gate g_i is good if we can easily find a set of gates $g_j, \dots, g_k$ such that the parity of the output of these gates is equal to the output of g_i, and thus, the evaluation of g_i can be reduced to the evaluation of $g_j, \dots, g_k$.

Proof of Theorem 1-5 The outline of the main steps of the proof is as follows: (1) show that there are very few bad gates; (2) construct a parity circuit D whose inputs are x and the bad gates, and whose interior gates are good gates; (3) show that for some assignment of values to the bad gates in D, the value of each gate in D is equal to the value of the corresponding gate in C; and (4) use the fact that D can be computed in polylog space.

Proof of Theorem 1-6

Claim 1 *The number of bad gates is at most e.*

Proof of Claim 1 Assume that there are $e+1$ bad gates and let $g_{h_1}, \dots, g_{h_{e+1}}$ be an enumeration of $e+1$ bad gates. Let $\mathcal{R}$ be the set of all nonempty subsets of $\{h_1, \dots, h_{e+1}\}$ of size at most e. Note for any $I \in \mathcal{R}$, that exactly one of $f(I\#0)$ or $f(I\#1)$ is in S, because exactly one of $I\#0$ or $I\#1$ is in A. So, let b_I be the unique $b \in \{0,1\}$ such that $f(I\#b) \in S$. Note also, for any distinct $I, J \in \mathcal{R}$, that $f(I\#b_I) \neq f(J\#b_J)$. Otherwise, g_k with $k = max(I \bigtriangleup J)$, which is bad by our assumption, is good, a contradiction. Since there are $2^{e+1} - 2 \geq 2^e$ elements in $\mathcal{R}$, we can collect 2^e elements in S, which contradicts the assumption that there are less than 2^e elements in S we see as the image of f. This proves the claim.

Proof of Claim 1 □

Proof of Theorem 1-7 Now let $g_{h_1}, \dots, g_{h_q}$ be the enumeration of all bad gates and let $H = \{h_1, \dots, h_q\}$, where $q \leq e$. For each good g_i, let $(I(i), b(i), J(i), c(i))$ be the lexicographically minimum $(I\#b, J\#b)$ witnessing that g_i is good. I 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 $g_0, g_1, \ldots, g_n, g_{h_1}, \ldots, g_{h_q}$. I will fix the input to g_0 to 1.
- Each interior gate g_i in D computes the parity function, whose direct inputs are given as follows:
  - If $b(i) = c(i)$, then all g_j with $j \in (I(i) \triangle J(i)) - \{i\}$.
  - If $b(i) \neq c(i)$, then all g_j with $j \in (I(i) \triangle J(i)) - \{i\}$ plus g_0.

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

Proof of Theorem 1-8 I say that $v \in \{0,1\}^q$ is valid if the value assigned by v to each bad gate is equal to the value of the bad gate in $C\#x$; i.e., for all $t, 1 \leq t \leq q$, the t-th bit of v is equal to $C[x, h_t]$. It is obvious that there is a unique valid v.

Claim 2 *v is valid if and only if for every gate $g_i, i, 1 \leq i \leq m$, $C[x,i] = D[1xv, i]$.*

Proof of Claim 2 The implication from right to left is obvious. The other direction is proven inductively. First, note that $C[x,i] = D[1xv,i]$ holds for every bad gate g_i. Next, let g_i be a good gate and suppose that the claim holds for every direct input g_j of g_i in D. I have $f(I(i)\#b(i)) = f(J(i)\#c(i))$. So,

$$b(i) = \bigoplus_{j \in I(i)} C[x,j] \text{ if and only if } c(i) = \bigoplus_{j \in J(i)} C[x,j]$$

This implies

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

where $j_1, \ldots, j_k$ is an enumeration of all j such that $j \neq i$ and $j \in I(i) \triangle J(i)$. By our supposition, for each t, $1 \leq t \leq k$, $D[1xv, j_t] = C[x, j_t]$. Also, by definition, g_0 is among the direct inputs of g_i if and only if $b(i) \neq c(i)$, i.e., $b(i) \oplus c(i) = 1$. Thus, $C[x,i] = D[1xv,i]$. Hence, the claim holds for g_i. This proves the claim.

Proof of Claim 2 □

Proof of Theorem 1-9

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

- If g_{h_t} is a NOT gate in C with g_j as its direct input, then v_t is equal to $\neg(D[1xv, j])$.
- If g_{h_t} is an AND gate in C with g_j and g_k as its direct inputs, then v_t is equal to $D[1xv, j] \wedge D[1xv, k]$.
- If g_{h_t} is an OR gate in C with g_j and g_k as its direct inputs, then v_t is equal to $D[1xv, j] \vee D[1xv, k]$.

Claim 3 *v is valid if and only if for all t, $1 \leq t \leq q$, v is correct at t.*

Proof of Claim 3 The implication from left to right is obvious. To prove the other direction, suppose that v is not valid. Let t be the smallest i such that the i-th bit of v is not equal to the output of the gate of C on input x; i.e., t is the smallest i, $1 \leq i \leq t$, such that $v_i = D[1xv, j_i] \neq C[x, j_i]$. Since D is topologically sorted, by an argument similar to that in the proof of Claim 2, we have $D[1xv, k] = C[x, k]$ for all $k < j_t$. If v is correct at t, then v_t is equal to $C[x, j_t]$, a contradiction. So, v is not correct at t.

Proof of Claim 3 □

Proof of Theorem 1-10

The above claims suggest the following algorithm to reduce C to D with the unique valid v.

Step 1: For each interior gate of C, test whether it is good, and construct H, the set of all bad gates.

Step 2: For each $v \in \{0,1\}^q$, test whether v is valid by testing whether v is correct at all t, and if so, use the valid v to compute $D[1xv, m]$.

Claim 4 *The algorithm can be executed in $O(\log^2 l)$ space.*

Proof of Claim 4-1

Proof of Claim 4 Let M be a logspace machine that computes f. Note that $I \in \mathcal{T}$ is encoded as a string of length $O(e \log m) = O(\log^2 l)$. Given $I\#b$ and $J\#c$, test whether $f(I\#b) = f(J\#c)$ can be done by simulating M on $I\#b$ and M on $J\#c$ simultaneously to compare $f(I\#b)$ and $f(J\#c)$ bit by bit. Since M's output tape is certainly write-only, the comparison requires storing only the most recent output bit from each. More precisely,

M on $I\#b$ and M on $J\#c$ are simulated alternatively step by step. If one of the simulations outputs a new bit of f, then it is suspended until the other simulation produces a new bit of f or halts without outputting a new bit. If both produce new bits, then the bits are compared and, if they are different, it must be the case that the values of f are different. The comparison is therefore terminated. If only one simulation produces a new bit, then the two values of f obviously have different lengths, so the values are different and the comparison is terminated. If both simulations halt without producing any new bits, then since the bits that have been produced so far are the same, it must be the case that they have the same value. The amount of space expended by the simulations is $O(\log^2 l)$, the amount required to store $I\#b$ and $J\#c$, since M is logarithmic space-bounded.

Proof of Claim 4-2 To test whether an interior gate g_i is good, and if so, to compute $I(i)$, $b(i)$, $J(i)$, and $c(i)$, it suffices to test, by cycling through all possible $(I\#b, J\#c)$ in the lexicographic increasing order, whether $(I\#b, J\#c)$ witnesses that g_i is good. By the previous discussion, the amount of space required is $O(\log^2 l)$. There are at most e bad gates, so the amount of space required to store H, the set of all bad gates, is $O(e \log m) = O(\log^2 l)$, so, H can be computed in space $O(\log^2 l)$.

Proof of Claim 4-3 Note that, as we are developing an $O(\log^2 n)$ algorithm, there is not enough space to store the entire description of D. However, after obtaining H, each bit of the description of D is computable in $O(\log^2 l)$ space as follows: In order to determine the direct inputs to g_i, if either $i \leq n$ or $i \in H$, then g_i is an input gate of D, and so, has no direct inputs; otherwise, $I(i), b(i), J(i), c(i)$, which are computable in $O(\log^2 l)$ space, provide the list of direct inputs.

Proof of Claim 4-4 To test whether v is correct at t, since h_t is the t-th element in H and the type of g_{h_t} in C and its direct input(s) are determined from $C\#x$, it suffices to compute $D[1xv, j]$ for j such that g_j is a direct input to g_{h_t} in C. Since D is a parity circuit, the computation problem is solvable by Parity-CVP. Recall that I demand that the last gate of a circuit be the output. So, let D_j be the circuit constructed from D by making the connection of g_m identical to that of g_j. Then $D_j(1xv) = D_j[1xv, m] = D[1xv, j]$. Let N be a deterministic Turing machine that decides Parity-CVP in $O(\log^2 n)$ space. Since each bit of the description of D is computable in $O(\log^2 l)$ space, given j, each bit of the description of D_j is computable in the same amount of space. Thus, one can simulate N on $D_j\#1xv$ by keeping track of the position of N's input head. When N needs to read the k-th bit of its input, one has only to activate the algorithm to produce D to compute the k-th bit (by recording the number of bits produced so far and the current bit), where the bits for the m-th gate are computed from those for the j-th gate. Thus, $D[1xv, j]$ is computable in $O(\log^2 l)$ space, and therefore, whether v is valid can be tested in the same amount of space.

Proof of Claim 4-5 Once the valid v is discovered, since it is of length at most e, there is enough space to record it. Now we have only to compute $C[x, m]$ as $D[1xv, m]$ with the valid v. Again, we have only to simulate N while computing the bits of D on demand, which requires $O(\log^2 l)$ space. Hence, the whole process can be done in $O(\log^2 l)$ space. This proves the claim.

Proof of Claim 4 □

This completes the proof of the theorem.

Proof of Theorem 1 □

3-2 By a straightforward generalization of the proof, I obtain the following theorem.

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

4 Conclusion

4-1 I have given a solution to the Hartmanis conjecture on sparse complete sets for P by showing that P cannot have many-one-hard sets of low density via space-efficient reductions unless $\mathrm{P} \subseteq \mathrm{DSPACE}[\log^2 n]$. I note here that, by extending the technique in this paper, Cai and Sivakumar have recently resolved the conjecture by showing that sparse P-hard sets exist under logspace many-one reductions if and only if P = L [CS95]. The technique has been further extended to study the sparse P-hard set problem for more flexible reducibilities [CNS95, vM95]. A very interesting open question in this regard is whether P having sparse hard sets under logspace Turing reductions collapses P.

5 Acknowledgment

5-1 The author would like to thank Eric Allender, Jin-yi Cai, Lane Hemaspaandra, Ioan Macarie, D. Sivakumar, and Marius Zimand for enjoyable discussions, and anonymous referees for many invaluable comments.

References

[ÀJ93] C. Àlvarez and B. Jenner. A very hard log-space counting class. *Theoretical Computer Science*, 107:3–30, 1993.

[BDHM92] G. Buntrock, C. Damm, U. Hertrampf, and C. Meinel. Structure and importance of Logspace-MOD class. *Mathematical Systems Theory*, 25:223–237, 1992.

[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.

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

[CS95] J. Cai and D. Sivakumar. The resolution of a Hartmanis conjecture. In *Proceedings of the 36th Conference on Foundations of Computer Science*, pages 362–371, Los Alamitos, CA, 1995. IEEE Computer Society Press.

[GP86] L. Goldschlager and I. Parberry. On the construction of parallel computers from various bases of Boolean functions. *Theoretical Computer Science*, 43:43–58, 1986.

[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.

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

[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.

[PZ83] C. Papadimitriou and S. Zachos. Two remarks on the power of counting. In *Proceedings of the 6th GI Conference on Theoretical Computer Science*, volume 145 of *Lecture Notes in Computer Science*, pages 269–276, Berlin, 1983. Springer-Verlag.

[Sav70] W. Savitch. Relationships between nondeterministic and deterministic tape complexities. *Journal of Computer and System Sciences*, 4:177–192, 1970.

[vM95] D. van Melkebeek. On reductions of P sets to sparse sets. Technical Report TR95-06, Department of Computer Science, University of Chicago, Chicago, IL, August 1995.