AN ESSAY ABOUT RESEARCH ON SPARSE NP COMPLETE SETS

By

J. Hartmanis
and
S.R. Mahaney

TR 80-422

Department of Computer Science
Cornell University
Ithaca, New York 14853

AN ESSAY ABOUT RESEARCH ON SPARSE NP COMPLETE SETS

J. Hartmanis and S. R. Mahaney
Department of Computer Science
Cornell University
Ithaca, New York 14853

Abstract

The purpose of this paper is to review the origins and motivation for the conjecture that sparse NP complete sets do not exist (unless P = NP) and to describe the development of the ideas and techniques which led to the recent solution of this conjecture.

1. Introduction

The research in theoretical computer science and computational complexity theory has been strongly influenced by the study of such feasibly computable families of languages as P, NP and PTAPE. This research has revealed deep and unsuspected connections between different classes of problems and it has provided completely new means for classifying the computational complexity of problems. Furthermore, this work has raised a set of interesting new research problems and created an unprecedented consensus about what problems have to be solved before real understanding of the complexity of computations can be achieved.

In the research on feasible computations the central role has been played by the families of deterministic and nondeterministic polynomial time computable languages, P and NP, respectively [AHU, C, GJ, K]. In particular, the NP complete languages have been studied intensively and virtually hundreds of natural NP complete problems have been found in many different areas of applications [AHU, GJ]. Though we do not yet know whether $P \neq NP$, we accept today a proof that a problem is NP complete as convincing evidence that the problem is not polynomial time computable (and therefore not feasibly computable); a proof that a problem is complete for PTAPE is viewed as even stronger evidence that the problem is not feasibly computable (even though there is no proof that $P \neq NP \neq PTAPE$).

As part of the general study of similarities among NP complete problems it was conjectured by Berman and Hartmanis, for reasons given in the next section, that all NP complete problems are isomorphic under polynomial time mappings and therefore there could not exist (sparse) NP complete sets with considerably fewer elements than the known classic complete problems (e.g. SAT, CLIQUE, etc. [BH]).

When the conjecture was first formulated in 1975, the understanding of NP complete problems was more limited and several energetic frontal assaults on this problem failed. As a matter of fact, the problem looked quite hopeless after a considerable initial effort to solve it. Fortunately, during the next five years a number of different people in Europe and America contributed a set of ideas and techniques which recently led to an elegant solution of this problem by S. Mahaney of Cornell University [M].

The purpose of this paper is to describe the origins of the sparseness conjecture about NP complete sets, to relate the information flow about this problem and to describe the development of the crucial ideas that finally led to the proof that if sparse NP complete sets exist, then P = NP [M].

We believe that this is an interesting and easily understandable development in the study of NP complete problems and that there are some lessons to be learned about computer science research from the way this tantalizing problem was solved.

Furthermore, it is hoped that these results may provide a new impetus for work on the main conjecture that all NP complete sets are p-isomorphic.

2. Preliminaries and the Sparseness Conjecture

Let P and NP denote, respectively, the families of languages accepted by deterministic and nondeterministic Turing machines in polynomial time.

A language C is said to be *NP complete* if C is in NP and if for any other language B in NP there exists a polynomial time computable function f such that

$$x \in B \Leftrightarrow f(x) \in C.$$

The importance of the family of languages P stems from the fact that they provide a reasonable model for the *feasibly computable* problems. The family NP contains many important practical problems and a large number of problems from different areas of applications in computer science and mathematics have been shown to be complete for NP [AHU, C, BJ, K]. Since today it is widely conjectured that $P \neq NP$, the NP complete problems are believed not to be solvable in polynomial time. Currently one of the most fascinating problems in theoretical computer science is to understand better the structure of feasibly computable problems and, in particular, to resolve the P = NP question. For an extensive study of P and NP see [AHU, GJ].

A close study of the classic NP complete sets, such as SAT, the satisfiable Boolean formulas in conjunctive normal form, HAM, graphs with Hamiltonian circuits, or CLIQUE, graphs with cliques of specified size, revealed that they are very similar in a strong technical sense [BH]. Not only can they be reduced to each other, they are actually isomorphic under polynomial time mappings as defined below:

Two languages A and B, $A \subseteq \Sigma^*$ and $B \subseteq \Gamma^*$, are *p-isomorphic* iff there exists a bijection $f: \Sigma^* \to \Gamma^*$ (i.e. a one-to-one and onto mapping) such that

1. f and f^{-1} are polynomial time computable,
2. f is a reduction of A to B and f^{-1} is a reduction of B to A.

Further study revealed that all the "known" NP complete sets are p-isomorphic and that one could formulate (after a number of technical lemmas) a very simple condition for NP complete sets to be p-isomorphic to SAT in terms of two padding functions [BH].

__Theorem 1:__ An NP complete set B is p-isomorphic to SAT iff there exists two polynomial time computable functions D and S such that

1. $(\forall x,y)\ [D(x,y) \in B \Leftrightarrow x \in B]$

2. $(\forall x,y)\ [S \circ D(x,y) = y]$.

All the known NP complete sets have these padding functions and in most cases they are easy to find. A good example is SAT, for which y can easily be encoded in any given formula in terms of new variables which do not change the satisfiability of the formula [BH].

From these studies grew the conviction that all NP complete sets are p-isomorphic and this conjecture was explicitly stated in [BH].

Clearly, if all NP complete sets are p-isomorphic then they all must be infinite and therefore $P \neq NP$. Thus it was realized that this conjecture may be very hard to prove, but the possibility was left open that it may be easier to disprove it. One way of disproving the p-isomorphism conjecture is suggested by the fact that p-isomorphic sets have quite similar densities. To make this precise we define sparseness below:

A set B, $B \subseteq \Sigma^*$ is said to be __sparse__ if there exists a polynomial p(n) such that

$$| B \cap (\epsilon + \Sigma)^n | \leq p(n).$$

Thus p(n) bounds the number of elements in B up to size n.

It is easily seen that SAT and other known NP complete sets are not sparse (any set possessing the padding functions D and S is not sparse) and that a sparse set cannot be p-isomorphic to SAT. These considerations lead to the conjecture [BH] that there do not exist sparse NP complete sets (unless P = NP). In particular, it was conjectured that no set over a single letter alphabet say $B \subseteq a^*$, can be NP complete.

It is interesting to note that the p-isomorphism conjecture quickly leads to the sparse set conjecture and then to the innocuous looking conjecture that no language on a single letter alphabet could be NP complete. We return to this last conjecture in the next section, it was the first to be solved.

A more indirect motivation for the p-isomorphism conjecture comes from the suggested analogy between recursive and recursively enumerable languages and P and NP as their feasibly computable counterparts. This analogy becomes particularly intrigueing and suggestive when it is extended to the Kleene Hierarchy and the polynomial time hierarchy [S]. The NP complete sets correspond in this analogy to the r.e. complete sets, which are known to be the same as the creative sets and they are all recursively isomorphic. This suggests that by analogy the NP complete sets should be p-isomorphic, as conjectured in [BH].

Lastly, a sparse NP complete set would imply that the necessary information to solve NP problems can be condensed in a sparse set. In other words, the sparse set could be computed and then used as a polynomially long oracle tape to solve other NP complete problems. At the time of stating the sparseness conjecture this looked very unlikely, and now we know that it is not possible unless P = NP. For related results discussing the consequences of the existence of polynomial size circuits for the recognition of SAT, see [KL].

3. Sparse Ranges and SLA

The p-isomorphism and sparseness conjecture and the more specialized conjecture that no language on a single letter alphabet can be NP complete received a fairly wide exposure at conferences and journal publications in the United States and Europe [BH, HB1, HB2]. Unfortunately, in spite of different attempts, no progress was made on this problem for several years and it started to look like an interesting problem about NP complete sets which was not likely to be solved in the near future.

The situation changed suddenly when Piotr Berman from Poland submitted a paper "Relationships Between Density and Deterministic Complexity of NP-Complete Languages" to ICALP '78. In this paper, motivated by the sparseness conjecture, P. Berman considered the consequences of P-time reductions with sparse range, particularly NP complete subsets of a^*. One of the authors was on the program committee for ICALP '78 and the paper, which in its first version was not easy to understand, was studied at Cornell with great interest. After some effort, with the help of S. Fortune, we convinced ourselves that indeed P. Berman's result was correct. In retrospect it is surprising how elegant and simple P. Berman's proof is and why so many other people who had thought about this problem missed it.

The paper was, as it amply deserved, accepted for ICALP '78 and received considerable attention. Unfortunately, P. Berman did not attend ICALP '78 himself and the paper was read at the conference by Ron Book, who had also worked on single letter alphabet languages [BWSD].

We state P. Berman's Theorem below and outline a proof:

<u>Theorem 2</u>: a) If there is a P-time reduction with sparse range for an NP complete set, then P = NP.

b) If there is an NP complete subset of a^*, then P = NP.

Note carefully that P. Berman's hypothesis of part a) is that there is a reduction g so that $|\{g(x) : |x| \leq n\}|$ is polynomially bounded. Though his proof used CLIQUE as an NP complete problem, we will consider the SAT problem in our outline of the proof. Part b), of course, is immediate from part a).

<u>Proof</u>: Let g be a p-time reduction of SAT to a sparse range. We outline an algorithm to determine if a boolean formula $F(x_1,\ldots,x_n)$, is satisfiable (and if so, finds an assignment). The algorithm will search part of a binary tree of self-reductions of F. The root is $F(x_1,\ldots,x_n)$. Each node will correspond to F with certain variables instantiated by 0 or 1 as follows: if $F(b_1,\ldots,b_{i-1},x_i,\ldots,x_n)$ is at a node, then its offspring will be

$$F(b_1,\ldots,b_{i-1},0,x_{i+1},\ldots,x_n)$$

and

$$F(b_1,\ldots,b_{i-1},1,x_{i+1},\ldots,x_n).$$

We construct the tree depth first, computing a label g(F) at each formula F encountered. The algorithm determines that certain formulas, F, correspond to unsatisfiable formulas and their labels, g(F), are marked as follows: a leaf with formula 0 (i.e. FALSE) is marked unsatisfiable; if both offspring of a node are marked unsatisfiable then the label at that node is marked unsatisfiable also. When a label is marked unsatisfiable other nodes occurring with the same label are similarly marked.

A careful analysis shows that whenever a bottom-most node is selected, then either a satisfying assignment is found or a new value g(F) is marked unsatisfiable in examining the next n nodes of the tree. Thus, the running time is polynomial in the size of $F(x_1,\dots,x_n)$.

QED

A close inspection of this proof shows that no explicit use has been made of the fact that the set A is in NP. Thus we have actually proved:

Corollary 3: If SAT can be reduced to a sparse set, then P = NP.

Even more fully formalized, P. Berman's proof is quite simple, but it provided the first important step in the solution of the sparseness conjecture. We believe that in the solution of this problem interaction between different groups played an important role and that a solution of even a highly specialized conjecture, like the sparse range case, provided the necessary impetus for further work.

4. No Sparse CO-NP Complete Sets.

In the attempt to understand P. Berman's proof of the single letter case, Steve Fortune, who at that time was a graduate student at Cornell, noticed that in Berman's proof the negative answers yielded valuable information. When a formula F is found to be unsatisfiable, its label g(F) is marked; one never has to explore beneath any other node of the tree with the same label value. Furthermore, such negative answers can be found only polynomially often before the possible values from $g(\ SAT^c)$ are exhausted.

This insight lead S. Fortune to a proof that the complete sets in CO-NP cannot be reduced to sparse sets, if $P \neq NP$ [F].

__Theorem 4__: If a CO-NP complete set can be reduced to a sparse set S, then P = NP.

__Proof__: Applying the same tree search method as before, observe that only negative answers are propogated up the tree by conjunctive self-reducibility (i.e., a node is not satisfiable if and only if both sons are not satisfiable). Since only the negative answers are used to prune the tree search, the polynomial running time is preserved under this weaker hypothesis.

QED

For a casual observer of theoretical computer science research the above result may look artificial since it does not answer the sparseness question, but instead solves a strange new problem about complete sparse sets for CO-NP. On the other hand, this was a critical step, as will be seen, in the solution of the general sparseness conjecture for NP complete sets.

5. __The Census Function__

Early in 1980, while working on his Ph.D. dissertation under Juris Hartmanis at Cornell, Steve Mahaney observed that if the exact number of elements in a sparse NP complete set can be computed in polynomial time, then some very interesting consequences followed, as stated below [HM].

For a set S let the [illegible] function C_S be defined by

$$C_S(n) = | S \cap (\epsilon+\Sigma)^n | \quad .$$

Mahaney's observation leads to the following result.

__Theorem 5__: If there exists a sparse NP complete set S with a polynomial time computable census function, C_S, then

$$NP \doteq CO\text{-}NP.$$

__Proof__: We will show that under the hypothesis we can recognize the complement of S in nondeterministic polynomial time. Since S^c is complete for CO-NP this guarantees that NP = CO-NP.

Given a string w, compute the census function $C_S(|w|)=k$. Using a nondeterministic polynomial time machine guess k different sequences $w_1, w_2, \ldots, w_k$, such that $|w_i| \leq |n|$, for $i=1,2,\ldots,k$ and verify that they all are in S using the NP recognizer of S. If the guessing and verification succeeds then w is in S^c iff $w \neq w_i$, $i=1,2,\ldots,k$. Thus, S^c is in NP and therefore NP = CO-NP.

QED

Combining the above result with Fortune's theorem we get the following.

Corollary 6: If there exists a sparse NP complete set, S, with a polynomial time computable census function then P = NP.

Proof: From the previous theorem, under the hypothesis of the corollary, we get that NP = CO-NP. But then every set complete for NP is complete for CO-NP and then, because S is a sparse complete set for CO-NP, by Fortune's result we get that P = NP.

QED

Again, the assumption that we have a sparse NP complete set with an easily computable census function may appear like imposing unnatural and restrictive conditions just to be able to derive a result. Surprisingly, the careful exploitation of the census functions lead a step closer to the solution of the sparseness conjecture.

6. Solution of the Sparseness Conjecture

During the spring of 1980 Karp and Lipton made available to us a draft of their forthcoming SIGACT paper "Some Connections Between Nonuniform and Uniform Complexity Classes" [KL]. This paper investigates the consequences of having "advice functions" (or oracles) which give values that depend only on the length of the input to be decided. Karp and Lipton develop uniform algorithms that utilize the existence, but not the easy computability, of such advice.

Two results in that paper are relevant to the sparseness conjecture. The first considers the consequence of having a Turing reduction of SAT to a sparse set or, equivalently, the existence of polynomial size circuits to solve NP (see Discussion below). The second result considered advice functions that yield only O(log(n)) bits of advice for inputs of size n.

__Theorem 7:__ Suppose h(.) is an advice function for NP satisfying

1. for some c, $| h(n) | \leq c \log(n)$, and
2. there is a deterministic polynomial time algorithm using $c \log(n)$ bits of advice that correctly decides SAT with advice h(.).

Then P = NP.

The proof of this theorem shows that all potential values of the c log(n) bits can be examined and the correct answer determined uniformly in polynomial time.

The deciphering of the Karp and Lipton paper, though it did not deal directly with the sparseness conjecture, suggested to Mahaney a new approach to the sparseness conjecture which combined the previously developed methods and led to its solution.

The intuitive link between Theorem 7 and the sparseness conjecture is found in the census results (Theorem 5 and Corollary 6). The unnatural hypothesis of the census results was that the census function, $C_S(n)$, was easily computable. Instead, observing that $C_S(n)$ is bounded by a polynomial, we see that the census may be written in O(log(n)) bits. The census results suggest a method to construct an algorithm that uniformly tries potential values of the census.

The essence of Mahaney's idea is to apply a census-like method (without knowing the exact census) to a sparse NP complete set to construct a p-time reduction of a CO-NP set to the sparse set, and then to use a Berman-Fortune depth first search method to solve SAT. The lack of knowledge about the census function is overcome by trying all of the polynomially many values for the census function and proving that the incorrect values can either be detected or that they cannot give a wrong answer.

In the proof below the ignorance about the census function is overcome by constructing a pseudo-complement of the sparse NP complete set S. The pseudo-complement incorporates guesses about what the corresponding census is and it is used to construct the desired sparse set of labels for the depth first search.

The outline of the proof below is as follows: We first give an NP recognizer for the "pseudo-complement" of the sparse set S. A reduction of this set to the sparse set S is used to provide the sparse set of labels for SAT^c; however, the certain computation of this set of labels requires knowing the census of S. Finally, the depth first search is modified to determine satisfiability of a formula (without exact knowledge of how to generate the sparse set of labels for SAT^c).

For the following discussion let $S \subset \{0,1\}^*$ be a sparse complete set for NP. Let M_S be a nondeterministic polynomial time recognizer of S and let

$$C_S(n) = |\, S \cap (\epsilon+\Sigma)^n \,| \le p(n)$$

where $C_S(.)$ is the true census function of S, and p(.) is a polynomial that bounds the size of the census.

We begin by constructing a Turing machine to recognize the pseudo-complement of S in nondeterministic polynomial time. Inputs include a padding $\#^n$ and an integer k which is a possible value of $C_S(n)$. Define the non-deterministic recognizer M by the following procedure:

$M(\#^n,s,k)$:

Check $|s| \le n$; otherwise reject.

Check $k \le p(n)$; otherwise reject.

Guess $s_1,\ldots,s_k$ so that

i. for all i, $|s_i| \le n$.

ii. for all i and j, $i \ne j \Rightarrow s_i \ne s_j$.

iii. for all i, check that s_i is accepted by M_S, the recognizer of S.

iv. check that for all i, $s \ne s_i$.

Lemma 8: Let $|s| \le n$ and $k \le p(n)$. Then on input $(\#^n,s,k)$ the machine M will:

1. accept if $k < c(n)$;

2. reject if $k > c(n)$; and

3. if $k = C_S(n)$, then M accepts if and only if M_S rejects s.

__Proof of Lemma__: We show part 3. If M accepts, then it will have enumerated the elements of S up to size n, verified that they belong to S, and shown that s is distinct from these elements. Since k is the true census, M accepts if and only if s is not in S.

QED

Intuitively, for $k = C_S(n)$, M is a recognizer of S complement. Moreover, M accepts its language in non-deterministic polynomial time (the input $\#^n$ is a padding to ensure this).

We will require labelling functions for pruning tree searches. The following discussion shows how to construct such functions from the sparse set S and many-one reductions of L(M).

Since M is an NP machine and S is NP complete, there is a p-time many-one reduction

$$g: L(M) \to S$$

so that for some monotonic polynomial q(.), inputs to M of size n are reduced to strings of size at most q(n) (cf. [C] and [K]). Similarly, for the NP-complete problem SAT, there is a P-time many-one reduction

$$f: SAT \to S$$

and a monotonic polynomial r(.) bounding the increase in size.

Let F of size m be a formula to be decided and let $n = r(m)$. Then any formula F' occurring in the tree of all self reductions will have size $\leq m$ and f(F') will have size at most n. Regarding k as a possible value for $C_S(n)$, we define

$$L_{n,k}(F') = g(\#^n, f(F'), k)$$

which will be the labelling function.

__Lemma 9__: Let F be a formula of size m and let $n = r(m)$. Furthermore, let k =

$C_S(n)$ be the true census. Then the function

$$L_{n,k}(F')$$

for formulas F' of size at most m satisfies:

1. F' is not satisfiable if and only if $L_{n,k}(F)$ is in S;

2. The unsatisfiable formulas of size at most m are mapped by $L_{n,k}$ to at most

$$p(q(2n+c\log(n))) \leq p(q(3n))$$

distinct strings of S where c is a constant depending only on p(.).

Proof: Part 1 is immediate from Theorem 5. For part 2 observe that $2n+c\log(n) \leq 3n$ is a bound on the size of $(\#^n, f(F'), k)$. Applying p o q gives an upper bound on the census of strings that the triple could map to.

QED

We now know that a suitable labelling function exists for $k = C_S(n)$; but we do not know $C_S(n)$, the true census! The algorithm in the following theorem shows how we can try $L_{n,k}$ for all $k \leq p(n)$.

Theorem 10: If NP has a sparse complete set, then P = NP.

Proof: We give a deterministic procedure to recognize SAT. Let F be a formula of size m. Apply the following algorithm:

```
begin
For k = 0 to p(r(m)) do
    Execute the depth first search algorithm using
        labelling function: L_{n,k}(F')
        at each node F' encountered in the pruned search tree.
    If a satisfying assignment is found,
        then halt; F is satisfiable.
    If a tree search visits more than
        m + m * p(q(3 r(m))) internal nodes,
        then halt the search for this k.
    end;
F is not satisfiable;
end
```

The algorithm clearly runs in polynomial time since the loop is executed at most $p(r(m))$ times and each iteration of the loop visits a polynomially bounded in m number of nodes.

The correctness of the algorithm is established in the following result.

Lemma 11: If F is satisfiable, then for $k = C_S(r(m))$ the search will find a satisfying assignment.

Proof: By Theorem 5, this k gives a labelling function that maps the unsatisfiable formulas of size at most m to a polynomially bounded set. Fortune shows that the depth first search will find a satisfying assignment visiting at most

$$m + m * p(q(3r(m)))$$

internal nodes.

QED

It is interesting to note here that we have not computed the census: a satisfying assignment could be found with any number of k's; similarly, if no satisfying assignment exists, many of the trees could be searched but the tree with k =

$C_S(r(m))$ is not distinguished.

The method of conducting many tree searches is parallelled in the uniform algorithm technique by Karp and Lipton [KL]. They show that if NP could be accepted in P with log( ) advice, then P = NP. The census function might be compared to a log( )-advisor to the polynomial information in the set S.

It is not necessary to assume an NP recognizer for the sparse set: just that S is NP-hard.

Lemma 12: If S is sparse and NP-hard, then there is a set S# that is sparse, NP complete, and has a P-time reduction: SAT --> S# that is length increasing.

Proof: Let f: SAT $\to$ S be a p-time reduction and let # be a new symbol. Define f#: SAT $\to$ S# by

$$f\#(F) = f(F)\#^p$$

where $p = \max\{0, |f(F)| - |F|\}$. Clearly S# is sparse. The mapping f# reduces SAT to S#. Membership of s in S# is verified by guessing a satisfiable formula that maps to s and verifying satisfiability.

QED

Corollary 13: If NP is sparse reducible, then P = NP.

7. Discussion

Although the isomorphism results [BH] are the direct ancestors of the work discussed here, the concept of sparseness has another motivation as stated in the Introduction: Can a "sparse amount of information" be used to solve NP problems in polynomial time? The approach here assumes the information is given as a many-one reduction to a sparse set.

For Turing reductions, the information is given as a sparse oracle set. A. Meyer has shown that a sparse oracle for NP is equivalent to the existence of polynomial size circuits to solve NP [BH]. The recent work by Karp, Lipton and Sipser [KL] has shown that if NP has polynomial size circuits, then the polynomial time hierarchy [S] collapses to Σ_2^P. Their result is weaker than Theorem 10, but it also has a weaker hypothesis. It is an interesting open problem to determine if polynomial size circuits for NP implies P = NP.

Similarly, now that we know that sparse NP complete sets cannot exist unless P $\neq$ NP, it would be interesting to determine whether there can exist sparse sets in NP - P. By Ladner's result [L] we know that if P $\neq$ NP then there exist incomplete sets in NP - P; the proof of this result does not yield sparse sets and we have not found a way to modify it to yield sparse sets.

For a related study of the structure of NP complete sets, see [LLR]. In this paper Landweber, Lipton, and Robertson explore the possibility of having large gaps in NP complete sets.

Finally, it is hoped that the success in solving the sparseness conjecture will initiate a new attack on the p-isomorphism conjecture for NP complete sets.

In conclusion, it is interesting to see how many people have directly or indirectly worked and contributed to the solution of the sparseness conjecture, among them, referenced in this paper are L. Berman, P. Berman, R. Book, D. Dobkin, S. Fortune, J. Hartmanis, R. Karp, L. Landweber, R. Lipton, S. Mahaney, A. Meyer, M. Patterson, E. Robertson, A. Selman, M. Sipser, and C. Wrathall.

References

[AHU] Aho, A.V., Hopcroft, J.E., and Ullman, J.D., The Design and Analysis of Computer Algorithms, Addison-Wesley (1974).

[B] Berman, P. "Relationship Between Density and Deterministic Complexity of NP-Complete Languages," Fifth Int. Colloquium on Automata, Languages and Programming, Italy (July 1978), Springer-Verlag Lecture Notes in Computer Science Vol. 62, pp. 63-71.

[BH] Berman, L. and Hartmanis, J., "On Isomorphisms and Density of NP and Other Complete Sets," SIAM J. Comput., 6 (1977), pp. 305-322. See also Proceedings 8th Annual ACM Symposium on Theory of Computing, (1976) pp. 30-40.

[BWSD] Book, R., Wrathall, C., Selman, A., and Dobkin, D., "Inclusion Complete Tally Languages and the Hartmanis-Berman Conjecture."

[C] Cook, S.A., "The Complexity of Theorem Proving Procedures," Proc. 3rd Annual ACM Symposium on Theory of Computing, (1977) pp. 151-158.

[F] Fortune, S., "A Note on Sparse Complete Sets," SIAM J. Comput., (1979), pp. 431-433.

[GJ] Garey, M.R., and Johnson, D.S., "Computers and Intractability, A Guide to the Theory of NP-Completeness," W.H. Freeman and Co., San Francisco, 1979.

[HB1] Hartmanis, J., and Berman, L., "On Polynomial Time Isomorphisms of Complete Sets," Theoretical Computer Science, 3rd GI Conference, March, 1977, Lecture Notes in Computer Science, Vol. 48, Springer-Verlag, Heidelberg, pp. 1-15.

[HB2] Hartmanis, J., and Berman, L., "On Polynomial Time Isomorphisms of Some New Complete Sets," J. of Computer and System Sciences, Vol. 16 (1978), pp. 418-422.

[HM] Hartmanis, J., and Mahaney, S.R., "On Census Complexity and Sparseness of NP-Complete Sets," Department of Computer Science, Cornell University, Technical Report TR 80-416 (April 1980).

[K] Karp, R., "Reducibility Among Combinatorial Problems," in Complexity of Computer Computations (R.E. Miller and J.W. Thatcher, eds.), Plenum, New York (1972).

[KL] Karp, R.M., and Lipton, R.J., "Some Connections between Nonuniform and Uniform Complexity Classes," Proc. 12th ACM Symposium on Theory of Computing, (May 1980).

[L] Ladner, R.E., "On the Structure of Polynomial Time Reducibility," J. Assoc. Computing Machinery, Vol. 22 (1975), pp. 155-171.

[LLR] Landweber, L.H., Lipton, R.J., and Robertson, E.L., "On the Structure of Sets in NP and Other Complexity Classes," Computer Science Tech. Report 342 (December 1978), University of Wisconsin-Madison.

[M] Mahaney, S.R., "Sparse Complete Sets for NP: Solution of a Conjecture by Berman and Hartmanis," Department of Computer Science, Cornell University, Technical Report TR 80-417 (April 1980).

[MP] Patterson, M, and Meyer, A.R., "With What Frequency are Apparently Intractable Problems Difficult?", Laboratory for Computer Science, M.I.T. Tech. Report., February 1979.

[S] Stockmeyer, L.J., "The Polynomial-Time Hierarchy," Theoretical Computer Science Vol. 3, (1977), pp. 1-22.