Sparse Sets in NP-P: Exptime Versus Nexptime

J. Hartmanis*
V. Sewelson*
N. Immerman†

TR 83-544
November 1983

*Department of Computer Science
Cornell University
Ithaca, New York 14853

†Department of Mathematics
Tufts University
Medford, Massachusetts 02155

SPARSE SETS IN NP-P: EXPTIME VERSUS NEXPTIME

J. Hartmanis
V. Sewelson
Department of Computer Science
Cornell University
Ithaca, New York 14853

N. Immerman
Department of Mathematics
Tufts University
Medford, Massachusetts 02155

Abstract

This paper investigates the structural properties of sets in *NP–P* and shows that the computational difficulty of lower density sets in *NP* depends explicitly on the relations between higher deterministic and nondeterministic time-bounded complexity classes. The paper exploits the recently discovered ***upward separation*** method, which shows for example that there exist sparse sets in *NP–P* if and only if $EXPTIME \neq NEXPTIME$. In addition, the paper uses relativization techniques to determine logical possibilities, limitations of these proof techniques, and, for the first time, to exhibit structural differences between relativized *NP* and *CoNP*.

Introduction and Outline of Results

It is well known that if higher deterministic and nondeterministic complexity classes do not collapse then the corresponding lower classes are also distinct. This ***downward separation*** is easily obtained by padding arguments and it yields, for example, that $EXPTIME \neq NEXPTIME$ implies $P \neq NP$. In this paper we prove that for some important complexity classes there also exists an ***upward separation method*** which reveals that the structural properties of *P*, *NP*, and *PSPACE* are explicitly determined by relations between the corresponding higher (exponential) complexity classes. We investigate this strong and unexpected coupling of the structural properties of lower and higher complexity classes by the upward separation method and use relativization techniques to explore possible structural properties of NP sets as well as limitations of these proof techniques.

The original motivation for this work was the desire to understand better what makes the computational solution of problems in NP hard, provided $P \neq NP$. It is generally conjectured that $P \neq NP$ [AHU,GJ] and therefore, for example, it is thought to be very difficult to determine whether Boolean formulas in conjunctive normal form have satisfying assignments. Furthermore, there is a more explicit belief that the computational difficulty of finding satisfying assignments for Boolean formulas does not depend only on the existence of the aggregate of satisfiable Boolean formulas, SAT, but that there are "individual" instances of formulas for which it is hard to find satisfying assignments. In particular, we conjecture that there are syntactically simple, sparse subsets of Boolean formulas for which it cannot be decided in polynomial time whether they are satisfiable.

In this paper we show that lower density sets in NP and $PSPACE$ (but apparently not in $CoNP$) can be coded more compactly to yield denser sets in the correspondingly higher complexity classes. Therefore, lower density sets can exist in $NP-P$ if and only if the higher complexity classes do not collapse.

More precisely, a set A is said to be *sparse* if and only if it contains only polynomially many elements up to size n [Ha].

Theorem: There exists a sparse set in $NP-P$, $PSPACE-NP$, and $PSPACE-P$ if and only if, respectively, $NEXPTIME \neq EXPTIME$, $EXPSPACE \neq NEXPTIME$, and $EXPSPACE \neq EXPTIME$.

This result has the interesting implication that if $P \neq NP$ but $EXPTIME = NEXPTIME$ then any sparse set in NP is already in P. Furthermore, since $A \in NEXPTIME-EXPTIME$ implies that $TALLY(A)$ is in $NP-P$ [Bo, Bet], we conclude that there exist sparse sets in $NP-P$ if and only if there exist tally sets in $NP-P$. Later in this paper we will see that this property does not necessarily hold for $CoNP-P$.

It is interesting to note that it has been shown by C.B. Wilson [Wi, BWX] that there exist oracle sets A such that

$$P^A \neq NP^A \text{ and } EXPTIME^A = NEXPTIME^A .$$

Furthermore, it has been shown more recently by S. Kurtz [Ku] that there are oracle sets B such that $P^B \neq NP^B$ and $NP^B - P^B$ contains no sparse sets. Since our results hold for relativized computations as well, we now know that $EXPTIME^A = NEXPTIME^A$ if and only if there are no sparse sets in $NP^A - P^A$. This shows that the two relativization results mentioned above are equivalent.

The above results can be generalized to show that the collapse of deterministic and nondeterministic classes below exponential time forces the corresponding sets of higher density from NP into P and that the collapse starts bounding the computation time of SAT.

Theorem: There are no $\delta(n) = n^{\log n}$ (uniformly distributed) dense sets in $NP - P$ if and only if

$$\bigcup_{c \geq 1} NTIME[2^{c\sqrt{n}}] = \bigcup_{c \geq 1} TIME[2^{c\sqrt{n}}]$$

and if this happens then $SAT \in TIME[2^{c\sqrt{n}}]$.

An interesting question is whether the assumption that SAT can be computed in less than exponential time has direct structural implications on the higher complexity classes. We show that at least for relativized computations the computation speed of NP problems can be decoupled from structural properties of the higher complexity classes.

Theorem: There exists an oracle A such that

$$NP^A \subseteq TIME^A[n^{\log n}]$$

and

$$EXPTIME^A \neq NEXPTIME^A .$$

These upward separation results have other implications about the relations between complexity classes based on the fact that standard diagonalization methods can be "slowed down" to yield sparse sets which separate complexity classes. We state one such result.

Corollary: If $TIME[n^{\log n}] \subseteq NP$ then

$$P \neq NP,\ EXPTIME \neq NEXPTIME,$$

$$EEXPTIME \neq NEEXPTIME, \text{ etc.,}$$

as well as

$$P \neq PSPACE \text{ and } EXPTIME \neq EXPSPACE.$$

Conversely, if we could separate P from NP by a sparse set then we would have shown that $P \neq NP$ as well as $EXPTIME \neq NEXPTIME$, which may be a much harder task than just showing that $P \neq NP$. Therefore, we conjecture that the separation of P and NP will be first achieved by an indirect proof or a constructive proof yielding a set in $NP-P$ which is not sparse. For related ideas see [Ko, KM].

In this connection it is interesting to observe that until now the separations of nondeterministic complexity classes have been obtained with proofs by contradiction (translation lemmas), not by explicitly yielding the sets which separate the complexity classes [Co, Ib, Se]. Furthermore, our results show that Ladner's [La] delayed diagonalization construction of an incomplete set in $NP-P$, under the assumption that $P \neq NP$, cannot be modified to yield a sparse set unless $EXPTIME \neq NEXPTIME$.

A very interesting open problem is the existence of sparse sets in the polynomial time hierarchy, PH [GJ, St], under the assumption that $EXPTIME=NEXPTIME$. Or equivalently, if there are no sparse sets in $NP-P$ can there exist sparse sets in $PH-P$? Quite surprisingly, we show that for some relativized computations the collapse $EXPTIME=NEXPTIME$ does not imply the collapse of the exponential hierarchy, $EXPH$, and therefore even though there are no sparse sets in $NP-P$ there can be sparse sets in $PH-P$.

Theorem: There exists an oracle A such that $EXPTIME^A=NEXPTIME^A$ and $\Sigma_2^{E(A)} \neq NEXPTIME^A$. Therefore there are no sparse sets in NP^A-P^A but there are sparse sets in $\Sigma_2^{P(A)}-P^A$.

We conjecture that the collapse of $EXPTIME=NEXPTIME$ implies the collapse of the exponential time hierarchy, but the above relativization result suggests that this may be a very hard result to prove.

Another relativization result reveals a limitation of the upward separation method and for the first time shows the existence of structural differences between relativized NP and $CoNP$ sets.

We first observe that the upward separation results for NP relativize and we get the following generalization.

Corollary: For any A, there are sparse sets in NP^A-P^A if and only if there are tally sets in NP^A-P^A and this happens if and only if $EXPTIME^A \neq NEXPTIME^A$.

Quite surprisingly this is not the case for relativized $CoNP$ computations from which we can force out tally sets without forcing out all sparse sets. This clearly indicates that the upward separation method does not work for $CoNP$ computations.

Theorem: There exists an oracle B such that $CoNP^B-P^B$ contains sparse sets but $CoNEXPTIME^B=EXPTIME^B$ and therefore there are no tally sets in $CoNP^B-P^B$.

Sparseness Results

In this section we introduce the upward separation method and derive necessary and sufficient conditions for the existence of sparse sets in $NP-P$, $PSPACE-NP$, and $PSPACE-P$, respectively. We assume that the reader is familiar with the standard results and notation about deterministic and nondeterministic polynomial time computations [AHU, GJ].

We say that a set S, $S\subseteq\Sigma^*$, is *sparse* if and only if there exists a constant k such that

$$|S\cap(\epsilon+\Sigma)^n|\leq n^k+k$$

i.e. the number of elements in S up to size n is polynomially bounded in n. Let P and NP denote the deterministic and nondeterministic polynomial time acceptable languages, respectively. Let

$$EXPTIME=\bigcup_{c\geq 1} TIME[2^{cn}]$$

and

$$NEXPTIME=\bigcup_{c\geq 1} NTIME[2^{cn}].$$

Theorem 1: There exists a sparse set S in $NP-P$ if and only if

$$EXPTIME\neq NEXPTIME .$$

Proof: If $EXPTIME\neq NEXPTIME$ then without loss of generality, we can assume that there exists a set A, $A\subseteq\{0,1\}^*$,

$$A\in NEXPTIME-EXPTIME .$$

If we prefix each string in A by a 1 and interpret these strings as binary representations of integers, then we can convert A to tally notation:

$$TALLY(A)=\{a^n\mid n\in 1A\} .$$

Since A is in $NEXPTIME$ and not in $EXPTIME$

$$TALLY(A)\in NP-P .$$

Thus we see that under this assumption there exists a sparse set in $NP-P$ [Bo, BWX].

We now assume that *NEXPTIME=EXPTIME* and will show that then every sparse set S in *NP* is already in P.

Assume that all the elements of S are lexicographically ordered, $S = \{x_1, x_2, \ldots\}$. Then define

$$S' = \{\#n\#i\#j\#t\#d \mid (\exists\, x, x_1, x_2, \ldots, x_i, y_1, y_2, \ldots, y_j \in S)$$

$$[x_1 < x_2 < \ldots < x_i < x \leq y_1 < y_2 < \ldots < y_j,$$

$$|x| = n, |y_j| = n \text{ and the } t^{th}$$

$$\text{digit of } x \text{ is } d]\} .$$

Since the integers n, i, j, t in the strings in S' are represented in binary, we see that

$$S' \in NEXPTIME ,$$

because in this time for input $\#n\ \#i\ \#j\ \#t\ \#d$ the appropriate number of strings of length n or less can be guessed and verified that they are in S and that they satisfy the required conditions. But then, because of our assumption,

$$S' \in EXPTIME .$$

From this follows that for input x, $|x| = n$, in polynomial time we can compute the maximal $i + j = m_n$ from the strings

$$\#n\#i\#j\#1\#0 \text{ or } \#n\#i\#j\#1\#1\ ,\ i, j \leq n^k + k$$

(if strings of this type are not in S' then no x of length n is in S). Now each string in S is identified by a unique i and j such that $i + j = m_n$. Thus x is in S if and only if for some $i_o + j_o = m_n$ the strings

$$\#n\ \#i_o\ \#j_o\ \#t\ \#d \text{ in } S',\ 1 \leq t \leq n,$$

define x. Since there are only polynomially many possibilities, they can be checked in polynomial time and we see that S is in P, as was to be shown.

Note: For the sake of emphasizing the importance of the census function in the previous proof we outline a second proof of Theorem 1:

If *NEXPTIME=EXPTIME* then for a sparse set S in *NP* the set

$$S' = \{(n,i) | (\exists x_1 < x_2 < \ldots < x_i)$$

$$[x_j \in S, \ |x_j| \leq n, 0 \leq j \leq i]\},$$

is in *NEXPTIME* and therefore in *EXPTIME*. But then the set

$$S'' = \{(n, m_n) \mid m_n = \max\{i \mid (n,i) \in S'\}\}$$

is in *EXPTIME*. Thus for input n in *EXPTIME* we can compute m_n. From this follows that the set

$$S''' = \{(n,i,t,d) \mid \text{the } i^{th} \text{ element of } S \text{ has } n \text{ digits}$$

$$\text{and } t^{th} \text{ digit is } d\}$$

is in *EXPTIME*. To see this note that for input (n,i,t,d) we can compute m_n, and then guess m_n elements in S, verify that they are in S, and then verify that the i^{th} element in S has n digits and that the t^{th} digit is d. Since this computation can be done in *NEXPTIME* we know that S''' is in *EXPTIME*. But then S is seen to be in P. □

By similar reasoning we can derive related results for *PSPACE*.

Corollary 2: There are sparse sets in *PSPACE–NP* and *PSPACE–P* if and only if *EXPSPACE* $\neq$ *NEXPTIME* and *EXPSPACE* $\neq$ *EXPTIME*, respectively.

Quite surprisingly, we will show later that related results do not hold for relativized *CoNP* computations, whereas all above results hold for relativized computations.

It is interesting to note that it is possible to relativize computations so that $NP^A \neq P^A$ and $EXPTIME^A = NEXPTIME^A$, in which case there are no sparse sets in $NP^A - P^A$ [Ku, Wi].

Next we show that the previous result can be sharpened considerably. We say that a set S is ***polynomial time printable*** if and only if there is a k_0 such that all the elements of S, up to size n, can be printed by a deterministic machine in time $n^{k_0} + k_0$. Clearly, every polynomial time printable set is sparse and is in P.

Corollary 3: There exists a polynomial time printable set S such that $S \cap SAT \in NP-P$ if and only if *EXPTIME* $\neq$ *NEXPTIME* .

Proof: From the previous result we know that if there exists a sparse set in *NP–P* then *EXPTIME* $\neq$ *NEXPTIME*. Therefore, we just have to show that

$$EXPTIME \neq NEXPTIME$$

implies that the desired set S exists and that

$$S \cap SAT \in NP-P \ .$$

Let $A \in NEXPTIME-EXPTIME$ then $TALLY(A) \in NP-P$. Since $TALLY(A)$ is in NP it can be reduced to SAT by a one-to-one, length increasing polynomial time reduction g [BH, AHU]. This guarantees that

$$g[TALLY(A)] \text{ and } g(1^*)$$

are sparse sets and furthermore that $g(1^*)$ is polynomial time printable. Since g is a reduction of $TALLY(A)$ to SAT we know that

$$x \in TALLY(A) \Leftrightarrow g(x) \in SAT \ .$$

Therefore $g[TALLY(A)] \subseteq SAT \cap g(1^*)$ and if $g(1^t) \in SAT$ then $1^t \in TALLY(A)$, yielding $g[TALLY(A)] = SAT \cap g(1^*)$.

Finally, $g[TALLY(A)] \in NP-P$ since $TALLY(A) \in NP-P$. This completes the proof.

The existence of sparse polynomial time recognizable sets S such that $S \cap SAT \in NP-P$ was conjectured by D. Joseph. The above results show that sparse sets exist in $NP-P$ if and only if there exist polynomial time printable sets S such that $S \cap SAT \in NP-P$.

Corollary 4: There exist sparse sets in $NP-P$ if and only if there exist tally sets in $NP-P$.

Finally, it is easily seen that we have actually shown that $EXPTIME = NEXPTIME$ if and only if every sparse set in NP is polynomial time printable. Therefore if one could show that there are sparse sets in P which are not polynomial time printable it would follow not only that $P \neq NP$ but also that $EXPTIME \neq NEXPTIME$.

Other Densities

For the sake of completeness we extend our results to higher complexity classes. These results emphasize dramatically that if the higher deterministic and nondeterministic complexity classes do not collapse, then super sparse subsets of Boolean formulas are in $NP-P$, clearly showing that the difficulty of determining whether a Boolean formula is satisfiable does not depend on the aggregate of satisfiable formulas, but that "individual" cases of such problems are computationally hard.

We say that a set A, $A \subseteq \Sigma^*$, is ***super sparse*** if there exists a constant k such that

$$|A \cap (\epsilon + \Sigma)^n| \leq k[\log n] .$$

Theorem 5: There exist super sparse sets in $NP-P$ if and only if

$$EETIME = \bigcup_{r \geq 1} TIME[2^{2^{r+n}}] \neq$$

$$\bigcup_{r \geq 1} NTIME[2^{2^{r+n}}] = NEETIME .$$

Proof: Again the proof is easy one way. To prove the implication the other way, assume that a super sparse set S is in NP. Then the set

$$C = \{\#t\#i\# \mid \text{up to size } 2^t \text{ there exist } i \text{ elements in } S\}$$

is contained in

$$NTIME[2^{2^{r+n}}] .$$

For x, $|x| \leq 2^t$ the representation of the length of t and i, is bounded by $\log\log|x|$. Therefore in doubly exponential time in the length of the input string $\#t\#i\#$ we can guess the i strings in S and verify that they are in S. If

$$NTIME[2^{2^{r+n}}] \subseteq TIME[2^{2^{s+n}}]$$

then we can compute in deterministic double exponential time the maximal i_t of $\#t\#i_t\#$ in C. This gives the number of strings in S up to size 2^t. But then, a deterministic doubly exponential time machine can compute for input t the sequence of i_t strings in S up to length 2^t,

$$\#x_1\#x_2\#\ldots\#x_{i_t}\# .$$

From this it follows that for any x, $|x|=n$, a deterministic polynomial time machine can compute the same string and check if x is in S. Thus if

$$\bigcup_{r \geq 1} NTIME[2^{2^{r+n}}] = \bigcup_{r \geq 1} TIME[2^{2^{r+n}}]$$

then S is in P. This completes the proof.

The sparseness results can also be generalized to sets of greater than polynomial density and the corresponding collapse of deterministic and nondeterministic classes below exponential time. Furthermore, the collapse of these classes starts bounding the computation time of SAT. We illustrate these possible generalizations with the next result.

We say that the set A has density $\delta(n)=n^{\log n}$ if

$$|A \cap (\epsilon+\Sigma)^n| \leq \delta(n).$$

To prove the following result we need the assumption that the $\delta(n)$-dense sets are uniformly distributed. We conjecture that this assumption is not needed, but so far have not been able to eliminate it from the proof.

A set A of density $\delta(n)$ is ***uniformly distributed*** if and only if any interval of length $2^n/\delta(n)$ contains at most polynomially many elements of A up to size n, where an ***interval*** is any set of strings consecutive in the lexicographic ordering of Σ^*.

Please note that for A to be uniformly distributed, it is enough for each of the $\delta(n)$ canonical intervals that divide Σ^n equally to have only polynomially many elements of A. A string x of length n belongs to the canonical interval containing only those strings of length n with the same $\log(\delta(n))$ leading bits.

Theorem 6: There are no $\delta(n)=n^{\log n}$ uniformly distributed dense sets in $NP-P$ if and only if

$$\bigcup_{c\geq 1} NTIME[2^{c\sqrt{n}}] = \bigcup_{c\geq 1} TIME[2^{c\sqrt{n}}]$$

and if this happens then $SAT \in TIME[2^{c\sqrt{n}}]$.

Proof: To prove one direction, suppose that

$$\bigcup_{c\geq 1} NTIME[2^{c\sqrt{n}}] \neq \bigcup_{c\geq 1} TIME[2^{c\sqrt{n}}],$$

and let $A \subseteq \{0,1\}^*$ and

$$A \in \bigcup_{c\geq 1} NTIME[2^{c\sqrt{n}}] - \bigcup_{c\geq 1} TIME[2^{c\sqrt{n}}].$$

Define

$$A' = \{x\# 1^{2^{\sqrt{\lfloor x \rfloor}}-|x|} \mid x \in A\}.$$

Clearly $A' \in NP-P$.

Given

$$w = x\# 1^{2^{\sqrt{\lfloor x \rfloor}}-|x|}$$

of length n, $|x| = [\log n]^2$. Therefore, there are at most $2^{[\log n]^2}$ such w, and so A' must have density $\leq n^{\log n}$. Each of the canonical intervals has at most one element of A' since A' has at most one string of length n beginning with any sequence of $[\log n]^2$ bits. Thus

A' is uniformly distributed.

Conversely, suppose $S \in NP$ has density $\delta(n)=n^{\log n}$ and that each canonical interval has fewer than $n^{k'}+k'$ elements of S. Then the set

$$C = \{\#t\#l\#i \mid \text{in the } l^{th} \text{ interval of strings of length } t$$

$$\text{there are at least } i \text{ elements in } S\}$$

is in $NTIME[2^{k\sqrt{n}}]$, for some k. Since for x of length t, the representation of t, i, and l is bounded by $[\log t]^2$, and in time $t^k=2^{k[\log t]^2}$, we can guess the $i \leq t^{k'}+k'$ strings and verify that they are in the l^{th} interval and in S. A string x is in the l^{th} interval if and only if its first $\log(\delta(n))$ bits are l, hence it is easy to verify.

If $NTIME[2^{k\sqrt{n}}] \subseteq TIME[2^{c\sqrt{n}}]$ then in $TIME[2^{d'\sqrt{n}}]$ we can compute the maximal $i_{t,l}$ such that $\#t\#l\#i_{t,l} \in C$. This gives the number of strings of S of length t in the l^{th} interval. But then, a $TIME[2^{d''\sqrt{n}}]$ machine can compute, for input $\#t\#l\#$, the $i_{t,l}$ strings in S in the l^{th} interval. Hence, for any x of length t, a deterministic polynomial time machine can compute the strings of S of length n in the same interval as x and check if $x \in S$. Thus $S \in P$.

To see how the computation time of SAT is affected, observe that $NP \subseteq \bigcup_{c>1} NTIME(2^{c\sqrt{n}})$. □

The crucial points that make the above technique work are that when we encode the $\delta(n)$ dense set into shorter strings, we encode strings of length n into strings of length $\log(\delta(n))$ and that to verify the encoded set, we guess no more than n^k+k strings for some k. This is where the uniformity was required.

It is our belief and we conjecture that any $\delta(n)$ dense NP set S is uniformly distributed in the sense that given a string x of length n, we can guess $\log(\delta(n))$ bits of information from which in time n^k+k, we can nondeterministically compute an interval that contains only polynomially many elements of S, including x. If this uniformity conjecture is true, not only may we drop the uniformity condition from the above theorem, but we have an interesting connection between NP sets and Kolmogorov complexity with bounded computatioin time.

Separation Results

Next we show that the upward separation results have very strong implications under the assumption that NP contains a deterministic, time bounded complexity class above polynomial. If this happens, then standard diagonalization arguments, when they are slowed down, can produce sparse and supersparse sets in $NP-P$. Clearly this forces the higher complexity classes not to collapse.

Theorem 7: Let $T(n)$ be real-time computable and for all $k \geq 1$ let

$$\lim_{n\to\infty} \frac{n^k}{T(n)} = 0.$$

Then $TIME[T(n)] \subseteq NP$ implies that

$$P \neq NP,\ EXPTIME \neq NEXPTIME,$$

$$EEXPTIME \neq NEEXPTIME, \text{ etc.}$$

as well as

$$P \neq PSPACE,\ EXPTIME \neq EXPSPACE, \text{ etc.}$$

Proof: Since $T(n)$ is real-time computable the limit condition permits us to diagonalize over all deterministic polynomial time machines. Since this diagonalization process can be stretched out by rejecting large segments of elements before diagonalizing over the next machine, we see that deterministic diagonalization can yield sets of any desired, computable density. Therefore, if

$$TIME[T(n)] \subseteq NP,$$

we know that $NP \neq P$ and because of the arbitrarily sparse sets in $NP-P$, we also know that the higher complexity classes do not collapse, $EXPTIME \neq NEXPTIME$, $EEXPTIME \neq NEEXPTIME$, etc.

Since $NP \subseteq PSPACE$ the same reasoning yields the other separation results. □

The previous result can easily be generalized to other complexity classes.

Corollary 8: Let $T(n)$ be real-time computable and for all $k \geq 1$ let

$$\lim_{n\to\infty} \frac{n^k}{T(n)} = 0.$$

Then $TIME[T(n)] \subseteq PSPACE$ implies that $P \neq PSPACE$, $EXPTIME \neq EXPSPACE$, etc.

It is interesting to note that these methods do not extend directly to nondeterministic time bounded computations. For example, we do not know yet whether $NTIME[n^{\log n}] \subseteq PSPACE$ implies that $NEXPTIME \neq EXPSPACE$, since we do not know whether there are sparse sets separating $NTIME[n^{\log n}]$ from NP. It should be recalled that the usual way of separating nondeterministic complexity classes is by translation lemmas, which are proofs by contradiction [Co, Ib, Se].

It is also interesting to note that if we could show that $P \neq NP$ by any process which can be "stretched" to yield sparse and super sparse sets, then we would not only have shown that $P \neq NP$, but also that

$$EXPTIME \neq NEXPTIME$$

and

$$EEXPTIME \neq NEEXPTIME.$$

Clearly, one possibility of showing that not only $P \neq NP$ but that the higher deterministic and nondeterministic time computations are different, is suggested by Theorem 7. This would require showing that NP contains some (properly defined) deterministic time classes above polynomial. Unfortunately, at this time we do not believe that this is true. We strongly conjecture that NP contains P properly and that furthermore NP contains no deterministic time classes above P. We conjecture that the corresponding relations hold between P and $PSPACE$ and NP and $PSPACE$.

Relativization Results

In this section we explore further the consequences of the upward separation method and use relativization techniques to determine logical possibilities as well as possible limitations of our proof techniques. During the course of this research, relativization techniques have played a very useful role and yielded some quite unexpected results, strongly influencing this research.

From Theorem 1 we know that a collapse of $EXPTIME = NEXPTIME$ forces all sparse sets from NP into P, but it seems to have no direct effect on the deterministic computation speed of SAT. On the other hand, the collapse of lower deterministic and nondeterministic complexity classes puts nontrivial computation bounds on SAT. For example,

$$\bigcup_{c\geq 1} TIME[2^{c\sqrt{n}}] = \bigcup_{c\geq 1} NTIME[2^{c\sqrt{n}}]$$

implies that SAT is in $TIME[2^{c\sqrt{n}}]$.

An interesting question is whether this implication also goes the other way: i.e. whether the assumption that SAT has fast deterministic (nonpolynomial) computation time has any structural implications about the higher complexity classes. In the following we show that at least for relativized computations we can decouple the computation speed of problems in NP from the collapse of the higher deterministic and nondeterministic complexity classes. As with all relativization arguments, this does not imply that this holds for regular (not relativized) computations, but it shows that there are "worlds" for which this result holds and indicates that the result is most likely very hard to prove or disprove for regular computations.

Theorem 9: There exists an oracle A such that

$$NP^A \subseteq TIME^A[n^{\log n}] \text{ and } EXPTIME^A \neq NEXPTIME^A .$$

Proof: We will construct A in stages so that A is the disjoint union of the following two sets

1. $0\#x\#i\#1^{|x|^{\log|x|}} \in A$ iff $N_i^A(x)$ accepts and $|x| \geq 2^{\sqrt{i \log i}}$, where $\{N_i\}$ is an enumeration of all nondeterministic polynomial time oracle machines such that for each i, N_i runs in time $n^{\log i} + \log i$.

   $S. = \{1^n \mid (\exists y)[\,|y| = 2^n \text{ and } 1y \in A]\}$
   $\in NEXPTIME^A - EXPTIME^A$.

Condition 1 will assure that $NP^A \subseteq TIME^A[n^{\log n}]$, since if $L \in NP^A$ then $L = L(N_i^A)$ for some i and we have encoded in A all but a finite initial portion of L.

S of condition 2 will be constructed by diagonalization over the deterministic exponential time oracle machines to ensure the separation of $NEXPTIME^A$ and $EXPTIME^A$.

Let $\{E_i\}$ be an enumeration of all exponential time oracle machines where E_i runs in time $2^{n \log i}$.

Construction:

We add elements to A in stages. Each stage has two parts, one for each condition. No elements are to be in A except those explicitly mentioned.

Let l be the index of the last exponential time machine cancelled. Let n_l be the stage at which we cancelled E_l.

Stage 0: $A_0 = \phi$, $l = 0$, $n_l = 0$

Stage n: $A_n = A_{n-1}$

> Part 1: Run machines $N_1^{A_n}, \ldots, N_n^{A_n}$ on all inputs, x, where $2^{\sqrt{(n-1)log(n-1)}} < |x| \leq 2^{\sqrt{nlogn}}$. If $N_i^{A_n}(x)$ accepts then add $0\#x\#i\#1^{|x|^{log|x|}}$ to A_n.
>
> Part 2: If $2^{n_l logl} < 2^n$, run $E_{l+1}^{A_n}(1^n)$. If not, go on to the next stage. If $E_{l+1}^{A_n}$ accepts 1^n, then we want $1^n \notin L$. Since there are no strings in L of the form $1y$ where $|y| = 2^n$, we do nothing. If $E_{l+1}^{A_n}$ rejects 1^n, we need to add a string, $1y$, to A_n. So, we add to A_n some string $1y$, $|y| = 2^n$, that was not queried during E_{l+1}'s computation.

$$l = l + 1$$

$$n_l = n$$

End of construction.

Observe that for any n, no computation in stage n part 1 can query strings of length greater than $2^{logn\sqrt{nlogn}} + logn$ which is less than $2^n + 1$, the length of strings that could be added to A_n in part 2. Thus, part 2 of stage n does not interfere with part 1. Also note that in part 2 of stage n, a computation can query strings of length at most $2^{log(l+1)n} \leq 2^{nlogn} < 2^{nlogn} + nlogn + 5$. Since part 1 of stage n just added all the strings of length less than $2^{nlogn} + nlogn + 5$ that any part 1 might add, part 1 will not interfere with part 2.

In part 2 of any stage, since we run $E_{l+1}^{A_n}(1^n)$ only when $2^{n_l logl} < 2^n$, adding a string of length $2^n + 1$ at this stage will not affect earlier stages, which queried strings of length up to $2^{n_l logl}$. And since $E_{l+1}^{A_n}(1^n)$ can query at most $2^{log(l+1)n}$ strings and there are 2^{2^n} strings, y, of length 2^n, we can always find a string, 1y, to add to A_n. □

A fascinating and important open problem in this research area is the relation between the nonexistence of sparse sets in $NP-P$ and other parts of the polynomial time hierarchy. The main problem is whether the collapse $EXPTIME=NEXPTIME$ which forces all sparse sets from NP into P also forces all sparse sets from the polynomial time hierarchy [GJ, St] into P.

The importance of this problem is emphasized by the fact that many interesting sparse sets are in the polynomial hierarchy. For example, it has been observed by A.R. Meyer that there exist polynomial size circuits for SAT if and only if there exists a sparse oracle set S such that $SAT \subseteq P^S$ [BH]. Furthermore, from [KL] we know that if such an oracle S exists then S is in the polynomial time hierarchy and the hierarchy is therefore finite. If $EXPTIME=NEXPTIME$ would also force all sparse sets from the polynomial time hierarchy into P, then the existence of polynomial size circuits for SAT or, equivalently, the existence of a sparse complete set for NP under polynomial time Turing reducibility, would imply that $P=NP$.

We recall that the existence of sparse complete sets under many-one polynomial time reductions implies that $P=NP$ [Ma].

Another interesting open problem is whether $EXPTIME=NEXPTIME$ implies that for every sparse subset $S \subseteq SAT$, which we know is in P, we can find in polynomial time a satisfying assignment for F in S. If $EXPTIME=NEXPTIME$ forces all sparse sets from the polynomial time hierarchy into P, then we can easily determine the minimal satisfying assignment of F in S in polynomial time from a sparse set in Σ_2^P, which by assumption is in P.

To define the *exponential hierarchy, EXPH*, let

$$\Sigma_1^E = NEXPTIME \text{ and } \Pi_1^E = CoNEXPTIME.$$

Σ_2^E consists of all languages C such that there exist a constant c and a polynomial time predicate R_c for which

$$C = \{x \mid (\exists y, |y| \leq 2^{c|x|})(\forall z, |z| \leq 2^{c|x|})[R_c[x,y,z]]\};$$

the other classes are defined analogously.

Equivalently, we can define the *exponential hierarchy* as follows:

$$\Sigma_0^E = EXPTIME \text{ and } \forall k > 0\, \Sigma_k^E = NEXPTIME^{\Sigma_{k-1}^P}.$$

Thus, for example, $\Sigma_2^P = NP^{SAT}$ and $\Sigma_2^E = NEXPTIME^{SAT}$. From this definition it is clear that a collapse of the polynomial hierarchy will imply a collapse of the exponential hierarchy. Since there is an oracle A such that $P^A \neq NP^A$ but $EXPTIME^A = NEXPTIME^A$ [Wi], it becomes equally clear that it is possible for $EXPTIME = NEXPTIME$ but yet to have the exponential hierarchy exist at higher levels. The reason a collapse at the base of the polynomial hierarchy topples the entire structure is the close coupling of the oracle and the underlying machine, e.g. $\Sigma_2^P = NP^{NP}$. In light of this, it is the behavior of the polynomial hierarchy that is peculiar, and not that of the exponential hierarchy. We expect the exponential hierarchy to collapse if $EXPTIME = NEXPTIME$ only because the hierarchy with which we have the most familiarity has the accident of being a special case.

As the next result shows, there are oracles for which $EXPTIME^A = NEXPTIME^A$ but the exponential hierarchy does not collapse, thus not all the sparse sets in the polynomial hierarchy are forced into P^A.

Theorem 10: There is an oracle A such that

$$EXPTIME^A = NEXPTIME^A \text{ but } \Sigma_2^{E(A)} \neq NEXPTIME^A .$$

Proof: We will build A such that for each nondeterministic machine M_i running in time $2^{|x|^{\log i}}$ on input x, A will include the string $0\#M_i\#x\#^{2^{\log(i+1)|x|}}$ if and only if M_i^A accepts x. This will ensure that $EXPTIME^A = NEXPTIME^A$. At the same time we will make sure that

$$L2(A) \text{ is in } \Sigma_2^{E(A)} - EXPTIME^A, \text{ where}$$

$$L2(A) = \{n \mid (\exists u, |u| = n)(\forall v, |v| = n)[1uv \notin A]\}.$$

Construction:

Just before stage n all strings in A of length less than $2n$ have been determined and deterministic exponential time machines $D_1, \ldots, D_{n-1}$ do not accept $L2(A)$, where D_i runs in time $2^{(\log i)n}$.

At stage n part 1: consider in turn the nondeterministic machines $M_1, M_2, \ldots, M_n$ on inputs $x_1, x_2, \ldots, x_{n^{\log(n)}}$ of length $[\log n]^2$. If M_1^A accepts x_1 then put $0\#M_1\#x_1\#^{2^{\log(2)[\log n]^2}}$ into A and reserve for $\overline{A}$ the strings queried by M_1's computation whose membership in A had not yet been determined. At most $2^{\log(2)[\log n]^2} = n^{\log n}$ elements of length at least $2n$ are newly reserved for $\overline{A}$.

If M_1^A rejects x_1 then to preserve this rejection by reserving for all $\overline{A}$ all strings queried on all computation paths would determine the membership of too many strings. Therefore, we consider all the ways of adding one point per class to some of A's empty classes of the form:

$$C(u)=\{1uv \mid |u|=|v|\}, \text{ for } |u|\geq n .$$

If M_1^A accepts x_1 using any of these A's, then take one and fix the elements of A queried by one accepting computation.

Do this next for M_1 on $x_2,\ldots,M_1$ on $x_{n^{\log n}},\ldots,M_n$ on $x_{n^{\log n}}$. The total number of elements reserved for $\overline{A}$ is bounded by

$$n^{\log n}[n^{\log n \log(2)}+n^{\log n \log(3)}+\ldots+n^{\log n \log n}]$$
$$< n^{(\log n+1)^2} < 2^{n-1},$$

for sufficiently large n.

Part 2 of stage n: Now consider deterministic machine D_n^A's computation on input n. Reserve for $\overline{A}$ the at most $n^{\log n}$ elements of A that this computation queries. Now if $D_n^A(n)$ rejects then fine because n is in $L2(A)$. However, if $D_n^A(n)$ accepts then we must add one point per empty $C(u)$, $|u|=n$, to A such that strings in $\overline{A}$ are not touched. This is possible by counting and it does not affect any of the previously fixed computations by construction. □

The above results seem to indicate that there may be a fundamental difference between many-one NP-completeness and Turing NP-completeness. In other words, that the existence of polynomial size circuits for NP may not necessarily imply that $P=NP$ as does the existence of sparse many-one complete NP sets [Ma].

In response to this question, very recently, S. Kurtz [Ku] and, simultaneously, N. Immerman and S. Mahaney [IM], have shown that there exists an oracle A and a sparse oracle S such that

$$NP^A \neq P^A \text{ but } NP^A \subseteq (P^A)^S.$$

This shows that for these oracles the Karp-Lipton result is indeed different from Mahaney's result [Ma,KL], i.e. there are worlds in which there exist sparse NP sets complete under Turing reductions but no sparse set is complete under many-one reductions.

Clearly the strong assumption that the exponential hierarchy collapses to $EXPTIME$ or the even stronger assumption that $EXPTIME=EXPSPACE$ has interesting implications since it forces all sparse sets from PH into P.

Corollary 11: If $EXPH=EXPTIME$ then the existence of polynomial size circuits for NP and $PSPACE$ implies, respectively, that $NP=P$ and $PSPACE=P$.

Proof: From [KL] we know that the polynomial size circuits for NP and $PSPACE$ are in PH and since their descriptions form sparse sets, the hypotheses guarantees that they are in P and therefore $NP=P$ and $PSPACE=P$, respectively. □

The upward separation method works well for NP and $PSPACE$ computations for which we can code down sparse sets based on ***guessing, verifying,*** and ***counting***. At the same time, $CoNP$ computations do not have explicitly the ability to guess as do NP computations and therefore the upward separation method does not seem to apply to sparse sets in $CoNP$. Our next result shows that at least for some relativized computations this is indeed the case.

Theorem 12: There is an oracle A such that there are sparse sets in $CoNP^A - P^A$ but there are no sparse sets in $NP^A - P^A$.

Proof: We will construct A in stages so that

1. $S = \{x \mid (\forall y, |y| = |x|)[1xy \notin A]\} \in CoNP^A - P^A$ and is sparse.
2. $0\#x\#1^{2^{k|x|}} \in A$ iff $N^A(x)$ accepts where N is a nondeterministic exponential time oracle machine running in time 2^{kn} such that $L(N^A)$ is complete for NE^A for all oracles A.

Condition 1 will be achieved by diagonalization over polynomial time oracle machines. Please note that condition 2 will force $NEXPTIME^A = EXPTIME^A$ which will ensure that there are no sparse sets in $NP^A - P^A$, since Theorem 1 relativizes.

Let $\{M_i\}$ be an enumeration of all polynomial time oracle machines where M_i runs in time $n^{\log i} + \log i$.

Construction:

We add elements to A in stages. Each stage has two parts, one for each condition. No elements are to be in A except those explicitly mentioned. At the start of stage n, $M_1, M_2, \ldots, M_{n-1}$, do not accept S.

Stage 0: $A_0 = \phi$

Stage n: $A_n = A_{n-1}$

Part 1: We shall diagonalize over the polynomial time oracle machines, adding to S at most one string of each length, ensuring sparseness.

If

i. $2^{k[log(n+1)]^2} 2^{[log(n+1)]^2} [log(n+1)]^2 n + (n^{log(n+1)} + log(n+1))n < 2^n$

then find some x_0 of length n such that for no y of length n has the membership of $1x_0y$ been determined. Since the left hand side of condition i is the number of strings of length $2n+1$ whose membership in A or $\overline{A}$ is determined, and there are 2^n x's of length n, we can find such an x_0. Run $M_{n+1}^{A_n}$ on input x_0, reserving for $\overline{A}$ all strings queried whose membership has not yet been determined. If it accepts, we want $x_0 \notin S$. So we add some $1x_0y$, $|y| = |x|$, to A_n. Since we reserved for $\overline{A}$ at most $n^{log(n+1)} + log(n+1) < 2^n$ strings, we can find a free $1x_0y$. If it rejects, we want $x_0 \in S$. So, we reserve for $\overline{A}$ all $1x_0y$, $|y| = |x_0|$. Note that condition i holds almost everywhere, so we have achieved the diagonalization.

In addition, we want S to remain sparse, in spite of strings reserved for $\overline{A}$ in future stages. So, we shall make sure that the only strings in S are those put in by the above diagonalization. To do this, for each $x \neq x_0$ of length n, add one $1xy$ to A_n, guaranteeing that $x \notin S$. Note that for every $x \neq x_0$ of length n there is some y of length n such that the membership of $1xy$ has not been determined, since the left hand side of requirement i is the number of strings of length $2n+1$ whose membership in A or $\overline{A}$ is already determined and it is less than 2^n. S will be sparse since i is true almost everywhere.

Note: We queried strings of length up to $n^{log(n+1)} + log(n+1)$.

We reserve for $\overline{A}$ at most $(n^{log(n+1)} + log(n+1))n$ queried strings in part 1's of all n stages so far.

We add strings of length $2n+1$.

Part 2: Run N^{A_n} on inputs x such that $[log(n)]^2 \leq |x| < [log(n+1)]^2$. If $N^{A_n}(x)$ accepts, add $0\#x\#1^{2^{k|x|}}$ to A_n and reserve for $\overline{A}$ all strings queried on some accepting computation path whose membership in A had not yet been determined. If $N^{A_n}(x)$ rejects, see if we can, by adding to A_n, make it accept. If so, add those strings and $0\#x\# 1^{2^{k|x|}}$ to A_n and reserve for $\bar{A}$ those strings queried along one accepting path. This will free us from reserving for $\bar{A}$ all queried strings on all computation paths (which would be too many for the counting argument), and will guarantee that strings added in future part 1's will not affect $N^{A_n}(x)$. If we cannot make $N^{A_n}(x)$ accept, simply reserve $0\#x\# 1^{2^{k|x|}}$ for $\bar{A}$.

Note: We determine the membership of at most $2^{k[log(n+1)]^2}2^{[log(n+1)]^2}[log(n+1)]^2 n$ strings in part 2's of all n stages so far.

We add strings of length $2^{k[log(n+1)]^2}+[log(n+1)]^2+3$.

End of construction.

Observe that in part 1, we query strings shorter than those to be added in part 2 so part 2 does not interfere.

Note that part 2 adds strings as it goes along that are larger than N^{A_n} could query, so part 2 does not interfere with itself. Since we reserve for $\bar{A}$ the strings we queried, the strings added in future part 1's can not affect what is done here. □

Corollary 13: There is an oracle A such that $CoNP^A-P^A$ has sparse sets, but no tally sets.

Proof: The oracle A of Theorem 12 suffices because $CoNP^A-P^A$ contains sparse sets, but because $NEXPTIME^A=EXPTIME^A$ all tally sets in NP^A are in P^A. Since tally sets have an easy syntax, this also forces all tally sets from $CoNP^A$ into P^A. □

This theorem has many interesting implications. It is the first oracle, A, to display a structural difference between NP^A and $CoNP^A$. Not only does it show that NP^A and $CoNP^A$ can be distinguished by the existence of sparse sets, but that only for $CoNP^A$ can we decouple sparse sets from tally sets. Note that by the same methods we can show structural differences for relativized *CoNP* and *PSPACE*, since the upward separation method works for *PSPACE* (see Corollary 2). It also demonstrates that the proof technique of the first theorem is in some sense tight, since when the technique is applied to *CoNP*, we must go to Σ_2^E, not *CoNEXPTIME* to decode the encoded sparse set. And since $EXPTIME=NEXPTIME$ does not necessarily imply that $\Sigma_2^E=EXPTIME$ the *CoNP* analog of the first theorem fails.

Acknowledgements

The authors would like to thank Ron Book, Deborah Joseph, Stuart Kurtz, Steve Mahaney, and Yaacov Yesha for helpful discussions and suggestions. The original search for the upward separation method was stimulated by Kurtz's relativization result [Ku] obtained during the AMS Summer Institute on Recursive Function Theory at Cornell University.

References

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

[BGS] T. Baker, J. Gill, and R. Solovay, "Relativizations of the $P=?NP$ Question", SIAM J. Computing (1975), 431-442.

[Bet] R.V. Book, et al, "Inclusion Complete Tally Languages and the Hartmanis-Berman Conjecture", Math. Systems Theory 11 (1978), 1-8.

[BH] L. Berman and J. Hartmanis, "On Isomorphisms and Density of NP and Other Complete Sets", SIAM J. Computing 6 (1977), 305-327.

[Bo] R.V. Book, "Tally Languages and Complexity Classes", Information and Control 26 (1974), 186-193.

[BWX] R.V. Book, C. Wilson, and M. Xu, "Relativizing Time and Space", IEEE-FOCS Symposium (1981), 254-259.

[Co] S.A. Cook, "A Hierarchy of Nondeterministic Time Complexity", Journal of Computer and System Sciences 7 (1973), 343-353.

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

[Ha] J. Hartmanis, "On Sparse Sets in NP-P", Department of Computer Science, Cornell University, TR82-508, August 1982.

[Ib] O. Ibarra, "A Note Concerning Nondeterministic Tape Complexities", Journal of the ACM 19 (1972), 608-612.

[IM] N. Immerman and S. Mahaney, "Oracles for Which *NP* Has Polynomial Size Circuits", draft, September 1982.

[KL] R.M. Karp and R.J. Lipton, "Some Connections Between Nonuniform and Uniform Complexity Classes", Proceedings 12th Annual ACM Symposium on Theory of Computation (April 1980), 302-309.

[KM] D. Kozen and M. Machtey, "On Relative Diagonals", IBM Research Report RC 8184, April 1980.

[Ko] D.C. Kozen, "Indexings of Subrecursive Classes", Proceedings 10th Annual ACM Symposium on Theory of Computing (1978), 287-295.

[Ku] S.A. Kurtz, "On Sparse Sets in $NP-P$: Relativization", to be published.

[La] R.E. Ladner, "On the Structure of Polynomial Time Reducibility", Journal of the ACM 22 (1975), 155-171.

[Ma] S. Mahaney, "Sparse Complete sets for *NP*: Solution of a Conjecture of Berman and Hartmanis", Proceedings 21st IEEE Foundations of Computer Science Symposium (1980), 42-49.

[Se] J. Seiferas, "Techniques for Separating Space Complexity Classes", J. Computer and System Science 14 (1977), 73-99.

[St] L.J. Stockmeyer, "The Polynomial Time Hierarchy", Theoretical Computer Science 3 (1976), 1-22.

[Wi] C.B. Wilson, "Relativization, Reducibilities, and the Exponential Hierarchy", Technical Report No. 140/80, Department of Computer Science, University of Toronto, Toronto, Ontario.