INFORMATION AND CONTROL 70, 54–68 (1986)

The Density and Complexity of Polynomial Cores for Intractable Sets*

PEKKA ORPONEN

Department of Computer Science, University of Helsinki, SF-00250 Helsinki, Finland

AND

UWE SCHÖNING

Seminar für Informatik, EWH Koblenz, D-5400 Koblenz, West Germany

Let A be a recursive problem not in P. Lynch has shown that A then has an infinite recursive *polynomial complexity core*. This is a collection C of instances of A such that every algorithm deciding A needs more than polynomial time almost everywhere on C. We investigate the complexity of recognizing the instances in such a core, and show that every recursive problem A not in P has an infinite core recognizable in subexponential time. We further study how dense the core sets for A can be, under various assumptions about the structure of A. Our main results in this direction are that if $P \neq NP$, then NP-complete problems have polynomially nonsparse cores recognizable in subexponential time, and that EXPTIME-complete problems have cores of exponential density recognizable in exponential time.

1. INTRODUCTION

Questions related to the existence and distribution of instances that make a computational problem hard are of fundamental importance in computational complexity theory. Many natural intractable problems are known to have fairly large collections of instances on which the problem is feasibly decidable: for example, the satisfiability problem can be decided in polynomial time on formulas in disjunctive normal form, on Horn formulas (Henschen and Wos, 1974), and on formulas in conjunctive normal form with at most two variables per clause (Aspvall, Plass, and Tarjan, 1979). On the other hand, Meyer and Paterson (1979) have shown that unless $P = NP$, no polynomial time algorithm for testing satisfiability can be so

* This research was supported in part by the Emil Aaltonen Foundation, the Academy of Finland, the Deutsche Forschungsgemeinschaft, and the National Science Foundation under Grant MCS83-12472. Part of the work was carried out while the authors were visiting the Department of Mathematics, University of California at Santa Barbara.

0019-9958/86 $3.00

comprehensive as to cover all but only a sparse set of the possible formulas. (A set of problem instances is called (polynomially) sparse if there is a polynomial bound on the number of instances of each size it contains (Berman and Hartmanis, 1977).)

A very interesting notion in this context is that of a *polynomial complexity core*, introduced by Lynch (1975). Given a problem A, this is a collection C of instances of A that is "uniformly hard" to decide, in the sense that if M is any deterministic Turing machine deciding A and p is any polynomial, then M runs for more than $p(|x|)$ steps on all but finitely many of the instances x in C. Lynch proved that any recursive problem A not in P has an infinite polynomial complexity core. Recently, the general conditions needed for a complexity class to permit such a core theorem have been investigated by Even, Selman, and Yacobi (1985), and by Du and Book (in press).

In this paper we strengthen Lynch's basic result in two respects. First, her proof does not set any particular bound on the complexity of recognizing the instances in a core; she only argues that recursive problems not in P have infinite *recursive* cores. We show that infinite polynomial cores for recursive intractable problems always exist in the class EXPTIME. (Actually, even in the class DTIME($t(n)$) for any time constructible superpolynomial function t.) Second, we investigate how dense the core sets can be for intractable problems with certain structural properties, such as paddability or completeness for NP or EXPTIME.

We show that if $P \neq NP$, then any recursive problem that is NP-hard with respect to polynomial time many–one reducibility has cores that are not polynomially sparse. Again, some of these nonsparse cores are only of subexponential complexity. We also obtain nonsparse cores of subexponential complexity for all the paddable problems not in P. (Intuitively, a problem is paddable if there is a fast method for generating a large number of inessential variants of any given instance, so that the status of the instance with respect to the problem is not changed. Paddability is a common property of all the "natural" NP-complete problems (Berman and Hartmanis, 1977; Young, 1983).)

Finally, for recursive problems that are EXPTIME-hard with respect to polynomial time many–one reducibility, we prove the existence of polynomial cores of a certain "exponential density." For EXTIME-complete problems such cores can again be found in the class EXPTIME.

2. Preliminaries

We study decision problems coded as sets of strings over the alphabet $\Sigma = \{0, 1\}$. The length of a strong $x \in \Sigma^*$ is denoted $|x|$; similarly, the car-

dinality of a set A is denoted $|A|$. For a set $A \subseteq \Sigma^*$ and $n \geqslant 0$, A_n denotes the finite set $\{x \in A \mid |x| \leqslant n\}$. If some predicate holds for all but finitely many elements in a set, we say that it holds *almost everywhere* (a.e.) in that set.

As a model of computation we use the deterministic multitape Turing machines; for this and other standard definitions, we refer the reader to, e.g., (Hopcroft and Ullman, 1979). The set of strings accepted by machine M is denoted $L(M)$, and the number of steps M makes on input x is denoted $\text{time}_M(x)$; this number may also be infinite. A machine is *total* if it halts on all inputs. For any function t on the natural numbers, we define

$$\text{DTIME}(t(n)) = \{L(M) \mid \text{time}_M(x) \leqslant t(|x|)\}.$$

The complexity classes P and NP have their usual definitions, and the class EXPTIME is defined as

$$\text{EXPTIME} = \bigcup \{\text{DTIME}(2^{cn}) \mid c \geqslant 0\}.$$

A function t on the natural numbers is *time constructible* if there is a Turing machine M that on inputs of length n halts in exactly $t(n)$ steps. We often use the increasing *standard sequence* of polynomials $p_1, p_2, \ldots$, defined by $p_i(n) = n^i + i$. A function is *superpolynomial* if it majorizes every polynomial almost everywhere.

A set $A \subseteq \Sigma^*$ is (polynomial time many–one) *reducible* to a set $B \subseteq \Sigma^*$, denoted $A \leqslant_m^p B$, if there is a polynomial time computable function f on Σ^*, such that for all $x \in \Sigma^*$, $x \in A$ if and only if $f(x) \in B$. For a class of sets $\mathscr{C}$, we say that A is *$\mathscr{C}$-hard* if every $B \in \mathscr{C}$ is reducible to A, and *$\mathscr{C}$-complete* if moreover $A \in \mathscr{C}$. (The term "$\mathscr{C}$-hard" is also sometimes used with reference to polynomial time Turing reducibility. Here many–one reducibility is always intended.)

3. A Complexity Bounded Version of Lynch's Core Theorem

We define complexity cores via the following very useful technical notion: given a Turing machine M and a function t on the natural numbers, the set of *t-hard inputs for M* is

$$H(M, t) = \{x \in \Sigma^* \mid \text{time}_M(x) > t(|x|)\}.$$

3.1. Definition (Lynch, 1975). A set $C \subseteq \Sigma^*$ is a *polynomial complexity core* for a set $A \subseteq \Sigma^*$, if given any Turing machine M accepting A and any polynomial p, C is almost everywhere contained in $H(M, p)$.

Lynch (1975) proved that every recursive set not in P has an infinite

recursive polynomial core. The general conditions underlying this proof were investigated by Even, Selman, and Yacobi (1985), who extended the result to several other complexity classes. (Still more general results have recently been obtained by Du and Book, in press.) We, too, begin by reconsidering Lynch's theorem, and by giving a very simple and transparent proof of a nonconstructive version of it.

3.2. PROPOSITION. *Let A be a recursive set not in P. Then A has an infinite polynomial complexity core.*

Proof. Let $M_1, M_2,...$ be an (in general noneffective) enumeration of *all* the Turing machines accepting A, and let $p_1, p_2,...$ be the standard sequence of polynomials. We construct a core set C for A in stages as follows. First we search for the smallest string (in the lexicographical ordering) x_1, such that $x_1 \in H(M_1, p_1)$. Such a string exists by the fact that $A \notin P$, which implies that the acceptor M_1 cannot run in polynomial time everywhere. Next we search for the smallest $x_2 > x_1$ such that $x_2 \in H(M_1, p_2) \cap H(M_2, p_2)$. Again, such an x_2 must exist, because otherwise the machines M_1 and M_2 combined (say, operating in parallel) would yield a polynomial time decision procedure for A.

We continue in this fashion, always at stage k looking for the smallest string $x_k > x_{k-1}$ such that $x_k \in \bigcap_{i=1}^{k} H(M_i, p_k)$, and arguing that such an x_k must exist, because otherwise combining $M_1,..., M_k$ would show that $A \in P$. (In fact, each one of the intersections is infinite.) The resulting set $C = \{x_1, x_2,...\}$ is clearly infinite; we show that it is also a polynomial core for A. Let M be any Turing machine accepting A, and let q be any polynomial. Then there is an index i such that $M = M_i$, and an index j such that the polynomials in $p_1, p_2,...$ majorize q from p_j on. Choosing $k_0 = \max\{i, j\}$, we see that for $k \geqslant k_0$,

$$x_k \in H(M_i, p_k) \subseteq H(M_i, p_j) \subseteq H(M, q).$$

Hence C is almost everywhere contained in $H(M, q)$. ∎

Since the above proof is based on the noneffective enumeration of the machines that accept A, the resulting core in general is not recursive. However, we next show how to base the same proof on an effective enumeration of machines with only slightly weaker properties. This modification then yields recursive, even moderately easy to decide cores. The important details are contained in the following lemma.

3.3. LEMMA. *For every recursive set A there is an effective enumeration of Turing machines $M_1, M_2,...$ with the following properties:*

(i) *For every $i \geqslant 1$, either $L(M_i)$ equals A, or $L(M_i)$ is finite.*

(ii) *For every* $i \geqslant 1$, *if* $L(M_i)$ *is finite, then* $H(M_i, q)$ *is co-finite for every polynomial* q.

(iii) *For every total Turing machine* M *accepting* A *and every polynomial* q, *there is an index* i *such that* $L(M_i) = A$ *and for some polynomial* r, $H(M_i, r) \subseteq H(M, q)$.

Proof. Let A be a recursive set, and let M_A be some total Turing machine accepting A. Let $t(n) \geqslant 2^n$ be a time constructible function bounding the running time of M_A. (See, e.g. (Schöning, 1982) for the details of finding such functions.) Let $T_1, T_2, \ldots$ be an enumeration of all Turing machines, augmented with clocks that shut off their computation after $t(n)$ steps. The machine M_i in the desired enumeration operates as follows:

On input x:

for every y, $t(|y|) \leqslant |x|$ **do**
  test whether $(y \in L(M_A) \Leftrightarrow y \in L(T_i))$; (1)

if the condition is true of every tested y
then if $x \in L(T_i)$ **then** accept **else** reject
else go to an infinite loop. (2)

Claim (i) of the lemma can now be verified by observing that either $L(T_i) = L(M_A) = A$, in which case $L(M_i) = L(T_i) = A$, or there is some smallest y such that $L(T_i)$ differs from A at y. But the latter condition will be detected at line (1) on every input x with $|x| \geqslant t(|y|)$, and M_i will reject all these inputs by entering the loop at line (2). In this case, then, $L(M_i)$ will be finite. Also, in this case all inputs x with $|x| \geqslant t(|y|)$ will be "infinitely hard" for M_i, proving claim (ii).

To verify claim (iii), let M be any Turing machine accepting A, and let q be any polynomial. Let M' be the machine obtained by combining M and M_A to run in parallel, so that M' accepts or rejects whenever the first of M and M_A accepts or rejects. Since M_A is $t(n)$ time bounded, M' also operates in time $t(n)$. Moreover, because M' halts no later than M, $H(M', q) \subseteq H(M, q)$. Augmenting M' with a $t(n)$ clock does not change its operation in any essential way, so there will be an index i such that $L(T_i) = L(M') = A$ and $H(T_i, q) = H(M', q)$.

Now consider the machine M_i. On input x, the tests at line (1) can all be performed in time polynomial in $|x|$, because t is time constructible, $t(|y|) \leqslant |x|$, and since $t(n) \geqslant 2^n$, the number of y's tested is linear in $|x|$. Hence there is a polynomial r such that $H(M_i, r) \subseteq H(T_i, q) \subseteq H(M, q)$, as desired. ∎

It is only a small change to use the effective enumeration provided by Lemma 3.3 in the proof of Proposition 3.2, instead of the original one.

Condition (ii) of the lemma guarantees that the "slack" machines in the enumeration—the ones accepting finite sets—do not block the construction in the proof from proceeding through all stages. Condition (iii) shows that the constructed set C, which for every index i and polynomial r has the property $C \subseteq H(M_i, r)$ a.e., indeed is a core for A. Thus the revised proof yields infinite recursive cores for recursive sets not in P. With more advanced techniques—applying the "slow diagonalization" methods developed by Ladner (1975) and Landweber, Lipton, and Robertson (1981)—we can even have moderately low bounds on the complexity of cores.

3.4. THEOREM. *Let A be a recursive set not in P. Then A has an infinite polynomial complexity core in the class* DTIME($t(n)$), *where t is any time constructible superpolynomial function.*

Proof. Let A and t be as in the statement of the theorem. Let $M_1, M_2, \ldots$ be the enumeration of machines provided by Lemma 3.3, and observe that for any standard encoding of the machines T_i, the mapping $i \mapsto M_i$ can be computed in polynomial time. Let $p_1, p_2, \ldots$ be the standard sequence of polynomials, and define a function r on the natural numbers as follows:

$$r(0) = 1; \quad \text{and for } k \geqslant 1,$$
$$r(k) = \text{the length of the smallest } x,\ |x| > k, \text{ such that } x \in \bigcap_{i=1}^{k} H(M_i, p_k) \text{ and } t(|x|) \geqslant k \cdot p_k(|x|).$$

It can be seen that r is a total recursive function: the machines M_i and polynomials p_k can be effectively enumerated, each of the sets $\bigcap_{i=1}^{k} H(M_i, p_k)$ is infinite, and for any k, $t(n)$ eventually majorizes the polynomial $k \cdot p_k(n)$. Let s be an increasing time constructible function majorizing r (cf. (Schöning, 1982)). For $m \geqslant 0$, denote by $s^{(m)}$ the m-fold composition of s, $s^{(m)}(x) = s(s(\ldots s(x)\ldots))$ (m times), and by I_m the mth *s-interval*:

$$I_m = \{x \in \Sigma^* \mid s^{(m)}(0) < |x| \leqslant s^{(m+1)}(0)\}.$$

Define a set C as

$$C = \bigcup_{m \geqslant 1} \left\{ x \in I_m \mid x \in \bigcap_{i=1}^{k} H(M_i, p_k) \text{ and } t(|x|) \geqslant k \cdot p_k(|x|) \text{ for } k = s^{(m)}(0) \right\}.$$

By the definition of r, and hence of s, every interval I_m contains at least one element in C, so C must be infinite. Also, it follows from the definition of C and condition (iii) of Lemma 3.3 that given any Turing machine M accepting A and any polynomial q, C is almost everywhere contained in $H(M, q)$. Hence C is a core for A.

Let us then consider the complexity of C. To determine whether an input

x is in C or not, first compute the unique $m \geqslant 0$ such that $s^{(m)}(0) < |x| \leqslant s^{(m+1)}(0)$. Since s is time constructible, this can be done in time polynomial in $|x|$ (cf. (Schöning, 1982)). Next check whether $t(|x|) \geqslant k \cdot p_k(|x|)$ for $k = s^{(m)}(0)$ and, if so, whether $x \in \bigcap_{i=1}^{k} H(M_i, p_k)$. If both conditions are satisfied, accept x, otherwise reject it. As t is time constructible, the first condition can be tested in time $O(t(|x|))$, and assuming it holds, the second one can be checked in time $O(k \cdot p_k(|x|)) = O(t|x|)$. It follows that $C \in \mathrm{DTIME}(t(n))$, as claimed. ∎

Even, Selman, and Yacobi (1985) observed that Lynch's theorem can be extended to show that every recursive set A not in P actually has an infinite recursive core *contained* in A. This extension can also be easily achieved by our methods: just use $H(M_i, p_k) \cap A$ instead of $H(M_i, p_k)$ in the proof of Proposition 3.1 or Theorem 3.4. With this modification, however, the techniques no longer necessarily yield core sets in $\mathrm{DTIME}(t(n))$ for any superpolynomial t. Since the cores are now constructed as intersections with A, we may only conclude that their complexity is $\mathrm{DTIME}(t(n))$ *relative to* A. (However, unrelativized time bounds in certain special cases have recently been obtained by Du (1985).)

Surprisingly, infinite polynomial cores may sometimes exist even in the class P. This is the case, for instance, with the so called *bi-immune* sets (Balcázar and Schöning, 1985). A set A is bi-immune if neither A nor $\bar{A}$ has infinite subsets in the class P. One of the basic properties of such sets established in (Balcázar and Schöning, 1985) is that they have all of Σ^*, which of course is a P-set, as a polynomial core.

Bi-immune sets exist in the class EXPTIME, but about their existence in lower complexity classes nothing is known. Interestingly, if the class NP contains such sets, then the NP-complete set SAT has an infinite polynomial core in P. To see this, assume that A is a bi-immune set in NP. Cook's proof of the NP-completeness of SAT (Cook, 1971) then yields a $\leqslant_m^p$-reduction f from A to SAT with the following simple property: given any $x \in \Sigma^*$, $f(x)$ is essentially just a code for the pair $\langle M, x \rangle$, where M is some fixed NP machine accepting A. In particular, for this reduction the image set $f(\Sigma^*)$ is in P. But $f(\Sigma^*)$ is also a core for SAT. Assume namely that there are infinitely many strings in $f(\Sigma^*)$ whose membership in SAT can be decided in polynomial time. Because f is a reduction, the infinitely many inverse images of these strings can be decided in polynomial time with respect to A, and so Σ^* cannot be a polynomial core for A.

4. Nonsparse Cores

The existence of an infinite core does not yet assert very much concerning "how" intractable a given set is: elements in the core can be very few

and far between. In this and the following section we show, however, that many intractable sets have not only infinite, but even relatively *dense* cores.

As a simple yardstick for assessing the density of a set we use the following notion due to Berman and Hartmanis (1977): a set $A \subseteq \Sigma^*$ is (polynomially) *sparse* if there is a polynomial p such that for every $n \geqslant 0$, $|A_n| \leqslant p(n)$. The technical notion of APT, or "almost polynomial time decidable" sets by Meyer and Paterson (1979) will also turn out to be very useful.

4.1. DEFINITION. The class APT consists of those recursive sets for which some decision procedure has only a sparse set of polynomially hard inputs, i.e.,

$$\text{APT} = \{A \mid A = L(M) \text{ for some total } M \text{ such that } H(M, p) \text{ is sparse for some polynomial } p\}.$$

The techniques used to prove Proposition 3.2, Lemma 3.3, and Theorem 3.4 provide a powerful set of tools for constructing core sets with prescribed properties. Here we apply these techniques to build nonsparse cores for sets not in APT. We begin with the analog of Proposition 3.2.

4.2. PROPOSITION. *Let A be a recursive set not in APT. Then A has a nonsparse polynomial complexity core.*

Proof. The construction follows the lines of Proposition 3.2, but instead of at each stage searching for a single string $x_k \in \bigcap_{i=1}^{k} H(M_i, p_k)$ to insert in the core, we look for a whole interval of strings at which the set $\bigcap_{i=1}^{k} H(M_i, p_k)$ is sufficiently dense. More precisely, we define a sequence of integers $n_0, n_1, n_2, \ldots$ as follows. First we set $n_0 = 0$, and then search for the smallest integer $n_1 > 0$ such that $|H(M_1, p_1)_{n_1}| > p_1(n_1)$. Such an integer exists, because otherwise the machine M_1 would witness that $A \in \text{APT}$. Next we search for the smallest $n_2 > n_1$ such that $|H(M_1, p_2)_{n_2} \cap H(M_2, p_2)_{n_2}| > p_2(n_2)$. Again, if such an n_2 did not exist, machines M_1 and M_2 could be combined to yield an APT procedure for A. And so on; in general at stage k we search for the smallest $n_k > n_{k-1}$ such that $|\bigcap_{i=1}^{k} H(M_i, p_k)_{n_k}| > p_k(n_k)$. Finally, we piece the core set together as

$$C = \bigcup_{k \geqslant 1} \left\{ x \in \Sigma^* \mid (n_{k-1_k} < |x| \leqslant n_k) \text{ and } x \in \bigcap_{i=1}^{k} H(M_i, p_k) \right\}.$$

It is straightforward to verify that C indeed is a core for A, and that for every polynomial p there is an n such that $|C_n| > p(n)$. ∎

This construction, too, can be made recursive by using the enumeration of machines from Lemma 3.3. Even a strong version similar to Theorem 3.4 is possible.

4.3. THEOREM. *Let A be a recursive set not in APT. Then A has a nonsparse polynomial core in the class* DTIME($t(n)$), *where t is any time constructible superpolynomial function.*

Proof. Similar to the proof of Theorem 3.4, but again considering intervals of strings instead of single strings. We only give the definition of the appropriate function r:

$$r(0) = 1; \text{ and for } k \geqslant 1,$$

$$r(k) = \text{the smallest } n > k \text{ such that}$$

$$\left| \bigcap_{i=1}^{k} H(M_i, p_k)_n \right| > p_k(n) \text{ and}$$

$$t(n) \geqslant k \cdot p_k(n). \quad \blacksquare$$

To apply Theorem 4.3 to some interesting classes of sets, we need results showing that some interesting classes of sets are disjoint from APT.

4.4. LEMMA (Meyer and Paterson, 1979). *Every set in APT is $\leqslant_m^p$-reducible to a sparse set.*

Proof. Let A be a set in APT, M a total machine accepting A, and p a polynomial such that the set $H(M, p)$ is sparse. Fix some $a \in A$ (we may assume that $A \neq \varnothing$). Then A can be reduced to the sparse set $(H(M, p) \cap A) \cup \{a\}$ by the following $\leqslant_m^p$-reduction f:

$$f(x) = \begin{cases} a & \text{if } M \text{ accepts } x \text{ in } p(|x|) \text{ steps;} \\ x & \text{otherwise.} \end{cases} \quad \blacksquare$$

We can now apply the well-known results about the nonreducibility of *NP*-complete sets to sparse sets (Berman, 1978; Fortune, 1979; Mahaney, 1982).

4.5. COROLLARY. *If $P \neq NP$, then every recursive NP-hard set (w.r.t. $\leqslant_m^p$-reducibility) has a nonsparse polynomial core in* DTIME($t(n)$), *where t is as in Theorem* 4.3.

Proof. If $P \neq NP$, then recursive *NP*-hard sets cannot be sparse (Mahaney, 1982), hence not in APT by Lemma 4.4. The claim now follows by Theorem 4.3. $\blacksquare$

Our second application of Theorem 4.3 relates to the notion of paddability (Berman and Hartmanis, 1977). A set $A \subseteq \Sigma^*$ is (polynomially) *paddable* if there exists a polynomial time computable 2-place function pad that is one-to-one and such that for all $x, y \in \Sigma^*$:

$$x \in A \Leftrightarrow \text{pad}(x, y) \in A.$$

Intuitively, the paddability of a set means that there is a fast way to generate a large number of different inessential variants of a given string, so that membership or nonmembership in the set is preserved. This idea is similar to the recursion theoretic notion of a "cylinder set" (Rogers, 1967, p. 89); it was introduced to the complexity context by Berman and Hartmanis (1977). They observed that all "natural" *NP*-complete sets have padding functions, even such that also the inverse function pad^{-1} can be computed in polynomial time. This kind of invertible paddability of an *NP*-complete set can be shown to imply that the set is polynomially isomorphic to *SAT*, and so Berman and Hartmanis conjectured that *all NP*-complete sets are polynomially isomorphic. Recently, however, Young (1983) has introduced a class of "structurally" defined *NP*-complete sets that do not seem to have padding functions.

As the issue of *NP*-completeness versus paddability remains unresolved, and also because there exist paddable sets not known to be either in *P* or *NP*-hard (e.g., ISO, the set representing the graph isomorphism problem), the following theorem is of certain interest.

4.6. THEOREM. *Every recursive paddable set not in P has a nonsparse polynomial complexity core in* DTIME($t(n)$), *where t is as in Theorem* 4.3.

Proof. We show that every paddable set in APT is already in *P*; the result then follows by Theorem 4.3. Assume that $A \in \text{APT}$ is paddable. Let M be a total machine accepting A and p a polynomial such that the set $H(M, p)$ is sparse. The idea of the proof is the following: given an input x, we can use the padding function of A to generate in polynomial time a number of variants of x that overwhelms the density of the hard-input set $H(M, p)$. Then at least one variant, and hence x, can be decided by M in time p.

More precisely, let q be a polynomial such that $|H(M, p)_n| \leqslant q(n)$ for every $n \geqslant 0$. Let pad be a padding function for A, and let r be a polynomial such that $|\text{pad}(x, y)| \leqslant r(|x| + |y|)$ for every $x, y \in \Sigma^*$. Let $y_1, y_2, \ldots$ be the strings in Σ^* in lexicographic order; note that in this ordering, $|y_i| \leqslant \log_2 i$ for every $i \geqslant 1$. We claim that for a sufficiently large polynomial s, the following polynomial time algorithm correctly decides A:

On input x:

for $i = 1$ **to** $s(|x|)$ **do**
begin
    run M on $\text{pad}(x, y_i)$ for $p(|\text{pad}(x, y_i)|)$ steps;
    if M accepts in this time **then** accept and halt
end;
reject and halt.

Let the algorithm be implemented by a Turing machine M'. If M' accepts an input x, then $\text{pad}(x, y_i) \in A$ for some i, and because pad preserves (non)membership in A, also $x \in A$. Hence $L(M') \subseteq A$. Conversely, if s is chosen so large that

$$s(n) > q(r(n + \log_2 s(n)))$$

for all n (choose s of higher degree than $q \cdot r$), then $A \subseteq L(M')$. This follows because if M accepts x, $|x| = n$, it accepts all strings of the form $\text{pad}(x, y_i)$. These are all different by the one-to-oneness of pad, and the first $s(n)$ have length at most $r(n + \log_2 s(n))$. Hence, M can run for more than $p(|\text{pad}(x, y_i)|)$ steps on at most $q(r(n + \log_2 s(n))) < s(n)$ many of them. ∎

The constructions in this section can again easily be modified to yield cores contained in the originating set A. It suffices to intersect the sets $H(M, p)$ by A throughout, and instead of the class APT consider the following asymmetric version:

$$\text{1-APT} = \{A \mid A = L(M) \text{ for some total } M \text{ such that } H(M, p) \cap A \text{ is sparse for some polynomial } p\}.$$

As a side effect we unfortunately again lose the complexity bounds except relative to A. (Though here, too, Du, 1985, has been able to reestablish unrelativized bounds in certain special cases.)

5. Polynomial Cores for EXPTIME-hard Sets

In this section we show that for EXPTIME-hard sets the core density results in the previous section can still be improved greatly. We say that a set $A \subseteq \Sigma^*$ has *exponential density* if there is a constant $\varepsilon > 0$ such that for almost every $n \geq 0$, $|A_n| \geq 2^{n^\varepsilon}$. Observe that if A has exponential density, then A is nonsparse in a very strong sense.

5.1. Theorem. *Let A be a recursive EXPTIME-hard set (w.r.t. $\leq_m^p$-reducibility). Then A has a polynomial complexity core $C \subseteq A$ of exponential density.*

Proof. Berman and Hartmanis (1977) showed how to construct a set $B \in \text{EXPTIME}$ with the following properties (a simpler construction is given in (Balcázar and Schöning, 1985)):

(i) every $\leq_m^p$-reduction f from B to some other set is almost everywhere one-to-one, i.e., there exist at most finitely many pairs (x, y), $x \neq y$, such that $f(x) = f(y)$; and

(ii) for every $n \geq 0$, $|B_n| \geq |\Sigma_n^*| - n = 2^{n+1} - 1 - n$.

Ko and Moore (1981) observed that a recursive set B satisfying clause (i) above is a polynomial core for itself. (In the terminology of Balcázar and Schöning, 1985, such a B is actually "strongly bi-immune".) This follows, because if some machine accepting B operated in polynomial time on an infinite set $E \subseteq B$, a reduction could be built to map all of E to a single string, violating (i). Then letting A be any EXPTIME-hard set, and f any reduction from B to A, $C = f(B) \subseteq A$ will be a polynomial core for A. This is because if some machine accepting A operated in polynomial time on an infinite set $E' \subseteq C$, a machine accepting B could be constructed to operate in polynomial time on the infinite set $f^{-1}(E') \subseteq B$, again contradicting the fact that B is a core for itself.

By property (i), the reduction f must be almost everywhere one-to-one. Let d be the number of pairs at which this one-to-oneness is violated, and let p be a polynomial bound on the time needed to compute f. Then for every $n \geqslant 0$,

$$|C_{p(n)}| \geqslant |B_n| - d \geqslant 2^{n+1} - 1 - n - d.$$

Hence for some $\varepsilon > 0$ depending on p (essentially the reciprocal of the degree of p), and almost every n,

$$|C_n| \geqslant 2^{n^\varepsilon}.$$

Thus C has exponential density. ∎

Let us consider the complexity of the core set produced above. Let A be a recursive EXPTIME-hard set; then there is a time constructible function t such that $A \in \mathrm{DTIME}(t(n))$. Let $B \in \mathrm{EXPTIME}$ be the "strongly bi-immune" set used in the proof. If f is any $\leqslant_m^p$-reduction from B to A, we may conclude that for almost every x in B, $t(|f(x)|) > |x|$: for otherwise, infinitely many inputs x in B can be decided in polynomial time by computing $f(x)$ and running a $t(n)$-bounded acceptor for A on this. From this observation it follows that the core $C = f(B)$ constructed in the proof is in the class $\bigcup\{\mathrm{DTIME}(2^{ct(n)}) \mid c > 0\}$. This is because the membership of an input y in the core can be decided by enumerating all the strings x of length less than $t(|y|)$, for each testing if $x \in B$, and if so, whether $f(x) = y$.

Consider then the special case of an EXPTIME-complete set A. The argument above shows that A has exponential density cores decidable in double exponential time. However, by using some recent results on the structure of EXPTIME-complete sets, we can imporve on this by one exponential.

5.2. THEOREM. Let A be an EXPTIME-complete set. Then A has a polynomial core of exponential density in EXPTIME.

Proof. Both Berman (1977) and Watanabe (1985) have shown that all EXPTIME-complete sets are interreducible by reductions that are one-to-one and length-increasing (i.e., such that $|f(x)| > |x|$ for every $x \in \Sigma^*$). Given B and A as above, let f_1 be an arbitrary reduction from B to A. Then f_2, given by $f_2(x) = \langle f_1(x), x \rangle$, is a length-increasing reduction from B to $A \times \Sigma^*$. As A is EXPTIME-complete, so is $A \times \Sigma^*$, and hence there is a length-increasing reduction f_3 from $A \times \Sigma^*$ to A. A length-increasing reduction f from B to A may now be obtained by defining

$$f(x) = f_3(f_2(x)) = f_3(\langle f_1(x), x \rangle).$$

If this particular reduction is used to define the core $C = f(B)$, the decision procedure sketched above works in single exponential time (i.e., with $t(n) = n$). ∎

6. Concluding Remarks

Polynomial complexity cores are collections of inputs that exhibit the inherent complexity of a problem in a "nonredundant" way, in the sense that every algorithm for the problem must be slow, i.e., run in non-polynomial time, almost everywhere on the core.

It has been known (Lynch, 1975) that all recursive intractable problems possess infinite recursive cores. We have improved on this by showing that recursive intractable problems have infinite cores of only subexponential complexity. Sometimes the cores may even be in P. An intriguing question that remains open is whether infinite cores always can be found in P. If this were the case, we could actually feasibly *decide* on some instances of, say, *SAT*, that they are "hard ones."

In a different direction, we have shown that cores for intractable problems can be moderately dense: nonsparse for NP-hard problems, and exponentially dense for EXPTIME-hard ones. These dense cores have also been shown to exist in moderately low complexity classes, though a question remains concerning the complexity of exponentially dense cores for sets that are EXPTIME-hard but not EXPTIME-complete.

All the core sets presented so far have been more or less artificially constructed. It would be extremely interesting to find *natural* examples of cores, which could then be viewed as natural "inherently intractable" subproblems. Exciting ideas in this direction have been presented (though without making the connection to cores explicit) by Krishnamurthy and Moll (1981), who have introduced a class of presumably hard tautologies, based on coding Ramsey numbers into propositional formulas.

ACKNOWLEDGMENTS

The authors would like to thank J. Balcázar, R. Book, T. Isákowitz,and D. Russo for their helpful comments on an earlier version of this paper, and J. Balcázar (again) and O. Watanabe for discussions in which the present version was greatly improved.

RECEIVED June 28, 1984; ACCEPTED January 21, 1986

REFERENCES

ASPVALL, B., PLASS, M. F., AND TARJAN, R. E. (1979), A linear time algorithm for testing the truth of certain quantified boolean formulas, *Inform. Process. Lett.* **8**, 121–123.

BALCÁZAR, J. L., AND SCHÖNING, U. (1985), Bi-immune sets for complexity classes, *Math. Systems Theory* **18**, 1–10.

BERMAN, L. (1977), "Polynomial Reducibilities and Complete Sets," Ph.D. thesis, Dept. of Computer Science, Cornell University.

BERMAN, L., AND HARTMANIS, J. (1977), On isomorphism and density of *NP* and other complete sets, *SIAM J. Comput.* **6**, 305–322.

BERMAN, P. (1978), Relationship between density and deterministic complexity of *NP*-complete languages, *in* "Proceedings, 5th Int. Colloq. on Automata, Languages, and Programming," Lecture Notes in Computer Science Vol. 62, pp. 63–71, Springer-Verlag, Berlin/New York.

COOK, S. A. (1971), The complexity of theorem-proving procedures, *in* "Proceedings, 13th Annu. ACM Sympos. on Theory of Computing," pp. 151–158, Assoc. Comput. Mach., New York.

DU, D. Z. (1985), "Generalized Complexity Cores and Levelability of Intractable Sets," Ph.D. thesis, Dept. of Mathematics, Univ. of California, Santa Barbara.

DU, D. Z., AND BOOK, R. V. (in press), The existence and density of generalized complexity cores, *J. Assoc. Comput. Mach.*

EVEN, S., SELMAN, A. L., AND YACOBI, Y. (1985), Hard-core theorems for complexity classes, *J. Assoc. Comput. Mach.* **32**, 205–217.

FORTUNE, S. (1979), A note on sparse complete sets, *SIAM J. Comput.* **3**, 431–433.

HENSCHEN, L., AND WOS, L. (1974), Unit refutations and Horn sets, *J. Assoc. Comput. Mach.* **21**, 590–605.

HOPCROFT, J. E., AND ULLMAN, J. D. (1979), "Introduction to Automata Theory, Languages, and Computation," Addison–Wesley, Reading, Mass.

KO, K. I, AND MOORE, D. (1981), Completeness, approximation and density, *SIAM J. Comput.* **10**, 787–796.

KRISHNAMURTHY, B., AND MOLL, R. N. (1981), Examples of hard tautologies in the propositional calculus, *in* "Proceedings, the 13th Annu. ACM Sympos. Theory of Computing," pp. 28–37, Assoc. Comput. Mach., New York.

LADNER, R. E. (1975), On the structure of polynomial-time reducibility, *J. Assoc. Comput. Mach.* **22**, 155–171.

LANDWEBER, L. H., LIPTON, R. J., AND ROBERTSON, E. L. (1981), On the structure of sets in *NP* and other complexity classes, *Theoret. Comput. Sci.* **15**, 181–200.

LYNCH, N. (1975), On reducibility to complex or sparse sets, *J. Assoc. Comput. Mach.* **22**, 341–345.

MAHANEY, S. R. (1982), Sparse complete sets for *NP*: Solution of a conjecture by Berman and Hartmanis, *J. Comput. System Sci.* **25**, 130–143.

MEYER, A. R., AND PATERSON, M. S. (1979), "With What Frequency Are Apparently Intractable Problems Difficult?," Tech. Rep. TM-126, Massachusetts Institute of Technology, Cambridge, Mass.

ROGERS, H., JR. (1967), "Theory of Recursive Functions and Effective Computability," McGraw-Hill, New York.

SCHÖNING, U. (1982), A uniform approach to obtain diagonal sets in complexity classes, *Theoret. Comput. Sci.* **18**, 95–103.

WATANABE, O. (1985), On one-one polynomial time equivalence relations, *Theoret. Comput. Sci.* **38**, 157–165.

YOUNG, P. (1983), Some structural properties of polynomial reducibilities, *in* "Proceedings, 15th Annu. ACM Sympos. Theory of Computing," pp. 392–401, Assoc. Comput. Mach., New York.