is not in P^A. Define

$$L_{ext} = \{1^n | \exists x \in A, |x| = n\}$$

L_{ext} is a tally set in NP^A, accepted by an NP^A machine N which given 1^n, nondeterministically chooses x of length n and inquires whether $x \in A$. Let $f(x)$ be the function computing the number of accepting computation of N on input x. Since $L_{odd} \in \mathrm{P}^{(\#\mathrm{P}_{(\delta)^A})}$, where $\delta(n) = 1$, we have $\mathrm{P}^{(\#\mathrm{P}_{(\delta)^A})} \neq \mathrm{P}^A$. Hence by the relativized versions of Theorems 7.4 and 7.2, for $h(n) \leq n - n^\epsilon$, there exists a language in P^A which does not have an h-prefix-P-ranker relative to A. However, since $\mathrm{P}^A = \mathrm{NP}^A$, all languages in NP^A have h-prefix-PTC's and h-prefix-PTD's relative to A. Thus we have the following.

Theorem 7.10 *There exists an oracle A such that*

1. *All languages in NP have h-prefix-PTC's relative to A for all prefixing functions h.*
2. *For each nondecreasing prefixing function h where $h(n) \leq n - n^\epsilon$ ($\epsilon > 0$), there exists a language L in P^A which does not have an h-prefix-P-ranker relative to A.*

References

[1] Theodore Baker, John Gill, and Robert Solovay. Relativizations of the P=?NP question. *SIAM Journal on Computing*, 4(4):431–442, December 1975.

[2] Juris Hartmanis, Neil Immerman, and Vivian Sewelson. Sparse sets in NP-P: EXPTIME versus NEXPTIME. *Information and Control*, 65:158–181, 1985.

[3] Lane A. Hemachandra and Steven Rudich. On the complexity of ranking. *Journal of Computer and System Sciences*, 41:251–271, 1990.

[4] Ker i Ko. Relativized polynomial time hierarchies having exactly k levels. *SIAM Journal on Computing*, 18(2):391–408, April 1989.

[5] Laura A. Sanchis and Mark A. Fulk. On the efficient generation of language instances. *SIAM Journal on Computing*, 19(2), April 1990.

[6] Alan A. Selman. A survey of one-way functions in complexity theory. *Mathematical Systems Theory*, 25:203–221, 1992.

[7] L. Valiant. The complexity of computing the permanent. *Theoretical Computer Science*, 8:189–201, 1979.

Proof: We only need to show that (3) $\Rightarrow$ (1). This follows combining the technique used to prove a similar statement in [3], and that used in the proof of Theorem 3.2. □

To examine the complexity of prefix ranking for the largest functions h we need another definition. Recall the definition of f-bounded NP machines and of the class $\mathrm{NP}^{cert(f)}$ from section 4. The function f is in the class $\#\mathrm{P}^{cert(f)}$ if it counts the number of accepting computations of an f-bounded NP machine.

Theorem 7.7 *Let h be a prefixing function such that $h(n) \geq n - n^{\epsilon}$ for some ϵ where $0 \leq \epsilon < 1$, for sufficiently large n. Let $g(n) = n - h(n)$ be nondecreasing. Then $P = P^{\#P^{cert(g)}}$ if and only if all languages in P have h-prefix-P-rankers.*

Proof: The proof of the *only if* direction is similar to the proof of Theorem 7.4. The other direction uses again a technique from [3]. Assume that all languages in P have h-prefix-P-rankers. Let L be a language in $\mathrm{NP}^{cert(g)}$ and N a machine accepting L, with computation paths of length $g(n)$. It suffices to prove that the number of accepting computations of N on an arbitrary input can be computed in polynomial time. Define L' to be the language of all strings of the form $x0^{h(n)}y$ where $|x| = n$, $|y| = g(n)$ and y is an accepting computation path for x in machine N, padded with additional 0's if necessary. Clearly L' is in P, and an h-prefix-P-ranker for L' can be used to count the accepting computations of N. □

In particular, it is easily seen that, as was the case for detectors and constructors, all languages in P have h-prefix-P-rankers when $h(n) \geq n - k \log n$.

Corollary 7.8 *Let h be a prefixing function such that $h(n) \geq n - n^{\epsilon}$ for some ϵ where $0 \leq \epsilon < 1$, for sufficiently large n. Let $g(n) = n - h(n)$ be nondecreasing. Then $P = NP$ and $P = P^{\#P^{cert(g)}}$, if and only if all languages in NP have h-prefix-P-rankers.*

Proof: The *only if* direction follows from the previous theorem. For the other direction, it suffices to prove that if all NP languages have h-prefix-P-rankers, then P=NP. But this follows from Theorem 3.2. □

Finally, we consider the relationships between the existence of h-prefix-PTD's, h-prefix-PTC's, and h-prefix-P-rankers. The only obvious implication is that if a language has either an h-prefix-P-ranker, or an h-prefix-PTC, then it must have an h-prefix-PTD. In particular, it appears feasible that particular languages or a class of languages might have h-prefix-PTC's without having h-prefix-P-rankers, and also viceversa. By Theorems 3.2 and 4.6, we do have the following.

Corollary 7.9 *Let h be a prefixing function such that $h(n) \geq n^{\epsilon}$ for some $\epsilon > 0$, for sufficiently large n. Then if all languages in P (NP) have h-prefix-rankers, then all languages in P (NP) have h-prefix-PTC's.*

In [4], Ko constructs an oracle A such that $\mathrm{P}^A = \mathrm{NP}^A$, and such that the language

$$L_{odd} = \{1^n | \text{ the number of strings of length } n \text{ in } A \text{ is odd } \}$$

Theorem 7.4 *Let h be a nondecreasing prefixing function such that $h(n) \leq n - n^{\epsilon}$ for some $\epsilon > 0$, for sufficiently large n. Let $\delta(n) = 2^{h(n)}$. Then $P = P^{\#P_{(\delta)}}$ if and only if all languages in P have h-prefix-P-rankers.*

Proof: Assume that all P languages have h-prefix-P-rankers, and let L be a $\delta(n)$ uniformly distributed dense set in NP. Let N be an NP machine accepting L, with computation paths of length at most $p(n)$. Assume without loss of generality that p is one-to-one. Choose integer $k \geq 1/\epsilon$. Define L' as follows. If x is the ith string of length n in L having prefix z ($|z| = h(n)$), and y is an accepting computation path for x on machine N (padded with additional 0's as necessary so that $|y| = p(n)$), put the string $z0^m y$ in L', where $h(n) + m + p(n) = \rho(i, p(n)^k)$. Note that $h(\rho(i, p(n)^k)) \geq h(n)$ and $h(\rho(i, p(n)^k)) \leq \rho(i, p(n)^k) - \rho(i, p(n)^k)^{\epsilon} \leq \rho(i, p(n)^k) - p(n)$. Since we are assuming that all languages in P have h-prefix-P-rankers, they all have h-prefix-PTD's, which implies by Lemma 3.12 and Theorem 4.1 that L is in P. Actually from the proof of Lemma 3.12 it may be seen that there is a polynomial-time procedure which given n and a prefix of length $h(n)$, will output all strings of length n in L having this prefix. It follows that L' is in P, and hence has an h-prefix-P-ranker, which implies that the function counting accepting computations of N is in P.

For the converse, assume that $\mathrm{P} = \mathrm{P}^{\#\mathrm{P}_{(\delta)}}$, and let L be any language in P. Define

$$L' = \{x1^{n-h(n)} \,||x| = h(n), \exists y, |xy| = n, xy \in L\}$$

L' is a $\delta(n)$ uniformly distributed dense set in NP, and has the following NP machine N: on input $x1^{n-h(n)}$, where $|x| = h(n)$, N nondeterministically chooses a string y of length $n - h(n)$ and determines whether $xy \in L$, accepting in this case. Because of the hypothesis, the number of accepting computations of N on any input can be computed in polynomial time (see similar proof of part of Theorem 7.1 in [3]). It follows that L has an h-prefix-P-ranker. □

Corollary 7.5 *Let h be a nondecreasing prefixing function such that $h(n) \leq n - n^{\epsilon}$ for some $\epsilon > 0$, for sufficiently large n. Let $\delta(n) = 2^{h(n)}$. Then if $P = P^{\#P_{(\delta)}}$ and P=NP, then all languages in NP have h-prefix-P-rankers.*

For large enough functions h within this range, a stronger statement can be made.

Theorem 7.6 *Let h be a nondecreasing function such that $n^{\epsilon} \leq h(n) \leq n - n^{\epsilon'}$ for some $\epsilon, \epsilon' > 0$, for sufficiently large n. Then the following are equivalent:*

1. *$P = P^{\#P}$.*
2. *All languages in NP have h-prefix-P-rankers.*
3. *All languages in P have h-prefix-P-rankers.*

of accepting computations of N on input x. The class $\mathrm{P}^{\#\mathrm{P}}$ consists of those languages which can be accepted by a polynomial-time machine with a #P oracle.

The following is proved by Hemachandra and Rudich:

Theorem 7.1 ([3]) *The following are equivalent:*

1. $P = P^{\#P}$.
2. *All languages in NP are strongly P-rankable.*
3. *All languages in P are strongly P-rankable.*

(The paper [3] also considers weaker ranking functions and shows that the existence of such weaker polynomial-time ranking functions for languages in P and NP is also equivalent to the above statements. Here we only consider the strong ranking functions).

We investigate a different notion of ranking function, which counts only the number of strings in a language satisfying a given prefix constraint.

Definition 7.1 Let h be a prefixing function. A language L has an h-prefix-P-ranker if there is a polynomial-time machine which given as input $(1^n, z)$, where $|z| = h(n)$, outputs the number of strings of length n in L which have prefix z.

It is clear that if a language is strongly P-rankable, then it has an h-prefix-P-ranker for any prefixing function h. Thus by Theorem 7.1, if $\mathrm{P} = \mathrm{P}^{\#\mathrm{P}}$, then all P and NP languages have h-prefix-P-rankers. We investigate whether the converse must hold. We show that the converse is true for h in a range extending between functions of the form n^ϵ and those of the form $n - n^{\epsilon'}$ $(0 < \epsilon, \epsilon' < 1)$. However, for both smaller and larger h, the question appears to be open and depends on the separation of certain counting-related complexity classes.

First note that by using the technique of Lemma 3.6, the following can be shown:

Theorem 7.2 *Let h_1, h_2 be prefixing functions such that h_1 is nondecreasing and $h_2(n) \leq h_1(n) \leq n - n^\epsilon$ for some ϵ, where $0 < \epsilon < 1$, for sufficiently large n. Then if all languages in P (NP) have h_1-prefix-P-rankers, then they all have h_2-prefix-P-rankers.*

Theorem 7.3 *Let h_1, h_2 be prefixing functions such that $n^\epsilon \leq h_1(n) \leq h_2(n)$ for some $\epsilon > 0$, for sufficiently large n. Suppose also that $n - h_1(n)$ is nondecreasing. Then if all languages in P (NP) have h_1-prefix-P-rankers, then they all have h_2-prefix–P-rankers.*

Thus we see that the existence of h-prefix-P-rankers for languages in P or NP is equivalent for all h in the range from n^ϵ to $n - n^{\epsilon'}$. It is within this range that the existence of h-prefix-P-rankers appears to be most unlikely, as we shall see.

We say the function f belongs to the class $\#\mathrm{P}(\delta)$ if there is a language L which is a $\delta(n)$ uniformly distributed dense set in NP, and an NP machine N accepting L, such that for each x, $f(x)$ is the number of accepting computations of N on x. One thing that characterizes functions in $\#\mathrm{P}(\delta)$ is that their values equal 0 for many inputs. For example, when $\delta(n)$ is a polynomial, the functions f in $\#\mathrm{P}(\delta)$ are such that $f(x) = 0$ for all but a polynomial number of strings x of length n.

exist P, NP, and NP-complete languages that have trivial h-PTC's. For L in P (or in NP, or NP-complete), define

$$L' = \{x0 | x \in L\} \cup \{y1 | y \in \Sigma^*\}$$

L' is in P, (or in NP, or NP-complete) and clearly has an h-prefix-PTC for each prefixing function h where $h(n) \leq n - 1$. When $h(n) = n$, all P languages have h-PTC's as well, whereas no NP-complete language can have one unless P=NP.

As mentioned in [5], most well-known P and NP languages appear to have length-based PTD's and PTC's. This apparently fits in with the conclusions reached in this paper, suggesting that 0-prefix-PTC's are in fact the easiest to obtain for NP languages. In addition, several well-known NP-complete languages seem to exhibit the pattern shown in these results. Consider for example the language SAT, consisting of encodings of satisfiable CNF formulas. Assume that each encoding begins with the number of variables m coded in binary, followed by an encoding for each clause, with some sort of demarcation between clauses. Note that for any $x \in SAT$, any prefix of x that is long enough can also be considered as a string in SAT, by completing if necessary the last (possibly incomplete) clause in the prefix. From this it follows that if $h(n) \geq n^\epsilon$, for some $\epsilon > 0$, then SAT cannot have an h-prefix-PTD unless P=NP (the proof is very similar to that of Theorem 3.2). It is not hard to see that SAT has a 0-prefix-PTC (this was shown in [5]) and by Theorem 3.8 this implies that SAT has an h-prefix-PTC whenever $h(n) \leq k \log n$. For values of $h(n)$ between $k \log n$ and n^ϵ, the same sorts of implications as shown in Theorem 3.14 can be shown: for example, if SAT has an h-prefix-PTD where $h(n) = \log^2(n)$, then SAT can be recognized in time $2^{c\sqrt{n}}$ for some c.

On the other hand there are other natural NP-complete languages which do not follow this pattern. Consider for example the language HAM consisting of encodings of hamiltonian graphs. Assume each encoding starts with the number of vertices t, followed by a bit string of length t^2 giving the adjacency matrix for the graph. Then HAM has an h-prefix-PTC where $h(n) = O(\sqrt{n})$ even if P $\neq$ NP. Nevertheless even for this language it appears that increasing h will tend to raise the difficulty of construction.

7 Ranking

In this section we consider a problem which is related to the construction and detection problems. Whereas a detector determines whether there exist any strings in a specified language having given characteristics, and a constructor produces such a string, a ranking function counts how many such strings exist. Hemachandra and Rudich have studied the complexity of ranking in [3] and related it to the counting class #P. If L is a language, a *strong rank function* $f_L : \Sigma^* \rightarrow N$ is defined in [3] as follows. For each $x \in \Sigma^*$, $f(x)$ equals the number of strings in L lexicographically smaller than or equal to x. A language is *strongly P-rankable* if its strong ranking function can be computed in polynomial time.

The class #P, defined by Valiant in [7], consists of counting functions $f : \Sigma^* \rightarrow N$. The function f is in #P if there is an NP machine N such that for each x, $f(x)$ equals the number

Proof: For each integer n and string z, where $|z| \leq h(n)$, define $s_n(z)$ to be the string of length exactly $h(n)$, containing z right justified with leading 0's as necessary. Let $S(x) = s_{|x|}(l(x))$. Let L be any language in P (NP). Define $L' = \{S(x)0^{n-h(n)}x | x \in L, |x| = n\}$. We assume by hypothesis that L' has an h-prefix-PTC M. To produce a string of length n with parameter value v, run M on input $(1^{2n}, s_n(v)0^{h(2n)-h(n)})$. □

If $h(n) \geq n^\epsilon$ for some $\epsilon > 0$, for sufficiently large n, then the existence of h-prefix-PTC's for all NP languages implies that P=NP, and hence the existence of all types of parameter-based PTC's for languages in NP.

Again we have the situation that for larger h the results for P languages are different from those for NP languages. Recall that for h strictly larger than $n - n^\epsilon$, it may be the case that all P languages have h-prefix-PTC's, even though not all NP languages can have such PTC's unless P=NP. (And in fact for $h(n) \geq n - k \log n$, all P languages must have h-prefix-PTC's). It turns out that the existence of h-prefix-PTC's for languages in P, for h in this range, does not necessarily imply the existence of parameter-based PTC's for parameter values of this size, as it does for smaller h.

Theorem 5.2 *Let h be a prefixing function such that $h(n) \geq n - n^\epsilon$ where $0 \leq \epsilon < 1$. Then there exists some parameter function l where $|l(x)| = h(|x|)$ for all $x \in \Sigma^*$, and a language $L \in P$, such that L does not have an l-PTC unless P=NP.*

Proof: Suppose that P $\neq$ NP. Then there exists a polynomial-time computable function $f : \Sigma^* \to \Sigma^*$ and polynomials q, r, such that $|f(x)| \leq q(|x|)$ and $|x| \leq r(|f(x)|)$ for all $x \in \Sigma^*$, and such that f cannot be inverted in polynomial time (see [6]). (This means that there cannot exist a polynomial-time computable function $g : \Sigma^* \to \Sigma^*$ such that if $g(y) = x$ then $f(x) = y$).

Assume without loss of generality that q is one-to-one. Define

$$L = \{x0^{2q(n)-n} | x \in \Sigma^*, |x| = n\}$$

L is clearly in P. Define l as follows. For $z \in \Sigma^*$, if $|z| = 2q(n)$ for some n, then $l(z) = f(x)0^{h(2q(n))-|f(x)|}$ where $x \sqsubset_n z$. Otherwise set $l(z) = 0^{h(|z|)}$. Note that

$$h(2q(n)) \geq 2q(n) - (2q(n))^\epsilon \geq q(n) \geq |f(x)|$$

for sufficiently large n. Thus $|l(z)| = h(|z|)$ for all $z \in \Sigma^*$. But if L had and l-PTC, then f could be inverted in polynomial time: given y, to find an x such that $f(x) = y$, it suffices to run the PTC on inputs of the form $(1^{2q(n)}, y0^{h(2q(n))-|y|})$ for all n up to $r(|y|)$. □

6 Discussion

It is worthwhile to point out that the results of the previous sections address the question of whether or not *all* P or NP languages have certain types of PTC's or PTD's. Individual languages in P or NP can behave quite differently. In particular, if $h(n) \leq n - 1$ then there

and therefore by hypothesis has an h-prefix-PTD. Note that $h(n^k) \geq n$ and $g(n^k) \geq g(n)$. It follows that L is in P. □

Thus the existence of h-prefix-PTD's and PTC's is equivalent for languages in P, for h in this range. A similar result follows for NP languages for $h(n) \geq n^\epsilon$ by Theorem 3.2. Thus the question of whether the existence of h-prefix-PTD's implies the existence of h-prefix-PTC's for languages in P or NP, is open only for prefixing functions h such that $h(n) < n^\epsilon$ for each $\epsilon > 0$, for infinitely many n.

5 Parameter-based construction

A more general way to view the construction problem is to base it on arbitrary parameter functions. Given a set of parameter functions, and corresponding parameter values, one wishes to find a string in a specified language having these values for the parameter functions. In this section we consider the existence of polynomial-time parameter-based constructors. In order to compare this type of construction with prefix-based constructors, we classify parameter functions by the number of bits necessary to encode their values. Since a constant number of parameter values can be encoded in a string whose length is a constant times the length of the longest single value encoding, we can without loss of generality restrict ourselves to constructors based on a single parameter function.

Since a parameter value for a given string x may in general be encoded in a number of bits which may be sublinear in the length of x, a parameter-based constructor that runs in polynomial time will also need as input some specification for the length of the desired string. In our definition we specify an exact length for the string, although it is not hard to see that similar results can be derived for more flexible length specifications, such as a lower and an upper bound.

Definition 5.1 Let $l : \Sigma^* \to \Sigma^*$ be a polynomial-time computable function, and q be a polynomial such that $|l(x)| \leq q(|x|)$. An l-PTC for a language L is a polynomial-time machine which given input $(1^n, v)$ either outputs a string x in L of length n, such that $l(x) = v$, or outputs the symbol Λ, indicating that no such string exists.

Prefix-based construction is of course just a special case of this type of construction. At first glance it may appear that the prefix function is simpler than other parameter functions. But in fact, for smaller h it can be easily shown that the existence of h-prefix-PTC's for all P or NP languages implies as well the existence of parameter-based PTC's for all parameter functions which can be encoded in at most $h(n)$ bits.

Theorem 5.1 *Let h be a nondecreasing prefixing function such that $h(n) \leq n^\epsilon$ for some ϵ where $0 < \epsilon < 1$, for sufficiently large n. Let $l : \Sigma^* \to \Sigma^*$ be a polynomial-time computable function such that $|l(x)| \leq h(|x|)$ for all $x \in \Sigma^*$. Then if all languages in P (NP) have h-prefix-PTC's, then they all have l-PTC's.*

Definition 4.1 Let $f : N \to N$ be a polynomial-time computable function bounded above by a polynomial. An NP machine N is f-bounded if each computation path of N on input x has length at most $f(|x|)$. We say a language L is in the class $\mathrm{NP}^{cert(f)}$ if it is accepted by an f-bounded NP machine.

In other words, a language L is in $\mathrm{NP}^{cert(f)}$ if it is in NP and has *certificates* of length $f(n)$ for strings of length n. It is obvious that $\mathrm{NP}^{cert(f_1)} \subseteq \mathrm{NP}^{cert(f_2)}$ whenever $f_1 \leq f_2$, and that any language in NP belongs to $\mathrm{NP}^{cert(f)}$ for some f. Also $\mathrm{NP}^{cert(f)} = \mathrm{P}$ if $f(n) \leq k \log n$. Moreover it is not difficult to check that if for each constant k, $f_2(n) - f_1(n) > k \log n$ for infinitely many n, then there exists an oracle separating the classes $\mathrm{NP}^{cert(f_1)}$ and $\mathrm{NP}^{cert(f_2)}$. We also have the following lemma.

Lemma 4.5 *Suppose that $f(n) \geq n^\epsilon$ for some $\epsilon > 0$. Then $P = NP$ if and only if $P = NP^{cert(f)}$.*

Proof: One way is clear. Assume that $\mathrm{P} = \mathrm{NP}^{cert(f)}$ and let L be in NP. Thus there exists a one-to-one polynomial p such that L has certificates of length $p(n)$. Let k be an integer such that $k \geq 1/\epsilon$ and let $r(n) = p(n)^k$. Define $L_1 = \{x0^{r(|x|)-|x|} | x \in L\}$. Clearly L_1 is in NP and has certificates of length $f(m)$ for strings of length m. Hence by hypothesis L_1 is in P, implying that L is in P. □

It is unclear whether a similar result holds for smaller functions f. It would appear that this is not the case, however we do not know of an oracle witnessing this hypothesis. The following theorem relates the collapsing of these classes into P, to the existence of PTC's and PTD's for languages in P.

Theorem 4.6 *Let h be a prefixing function such that $h(n) \geq n^\epsilon$ for some ϵ where $0 < \epsilon < 1$ for sufficiently large n. Let $g(n) = n - h(n)$ be nondecreasing. Then the following are equivalent.*

1. *$P = NP^{cert(g)}$.*
2. *All languages in P have h-prefix-PTC's.*
3. *All languages in P have h-prefix-PTD's.*

Proof: It suffices to prove (1) ⇒ (2) and (3) ⇒ (1).

(1) ⇒ (2): Assume that $\mathrm{P} = \mathrm{NP}^{cert(g)}$. Let L be a language in P. Define

$$L_1 = \{xy | |x| = |y| = n, x \approx_{h(n)} y, \text{ and } \exists z \in L \text{ such that } x \leq z \leq y\}$$

L_1 is in NP and has certificates of length $g(m/2) \leq g(m)$ for strings of length m. Hence by hypothesis L_1 is in P. An h-PTC for L can constructed by doing binary searches on L_1.

(3) ⇒ (1): Assume that all languages in P have h-prefix-PTD's. Choose integer k such that $k \geq 1/\epsilon$. Let L be in $\mathrm{NP}^{cert(g)}$. Define L_2 to consist of all strings of the form $x0^m y$ where $x \in L$, $|x| = n$, y is a certificate for x of length $g(n)$, and $|x0^m y| = n^k$. L_2 is in P

Proof: Let L be in NP. Thus there is a language L' in P, consisting of pairs $\langle x, y\rangle$ where $|y| = p(|x|)$ for some polynomial p, such that $x \in L$ if and only if $\langle x, y\rangle \in L'$ for some y. Let k be such that $p(n) \leq n^k$, and choose integer $k_1 \geq k/\epsilon$. Let $r(n) = \lceil 2^{1/\epsilon}\rceil n^{k_1}$. Define

$$\begin{aligned}\tilde{L} \;=\; & \{x_1 0^{m_1} x_2 0^{m_2} y | x = x_1 x_2 \in L, |x| = n, x_1 \sqsubset_{h(n)} x, \\ & \langle x, y\rangle \in L', m_1 = h(r(n)) - h(n), \text{ and } m_2 = r(n) - n - h(r(n)) + h(n) - p(n)\}\end{aligned}$$

Note $m_1 \geq 0$ since h is nondecreasing. We also have $r(n) - h(r(n)) \geq r(n)^\epsilon \geq 2n^k \geq p(n) + n$, and hence $m_2 \geq 0$. Clearly $\tilde{L}$ is in P, and the fact that $\tilde{L}$ has an h-prefix-PTC (PTD) implies that L has an h-prefix-PTC (PTD). □

Thus in particular, by Theorem 3.2, for $h(n)$ such that $n^{\epsilon_1} \leq h(n) \leq n - n^{\epsilon_2}$, for some $\epsilon_1, \epsilon_2 > 0$, h-prefix-PTC's or PTD's cannot exist for all languages in P unless P=NP. It remains to examine the existence of prefix-PTC's for languages in P, for larger functions h. We start with the largest prefixing functions, for which clearly all languages in P must have prefix-PTC's.

Theorem 4.2 *Let h be a prefixing function such that $h(n) \geq n - k\log n$ for some constant k for sufficiently large n. Then all languages in P have h-prefix-PTC's.*

Proof: To locate a string in L of length n, having prefix x of length $h(n)$, it suffices to try the $2^{n-h(n)} \leq n^k$ possible strings of the form xy where $|y| = n - h(n)$. □

In fact, the difficulty of construction appears to decrease as h increases, within this range, as suggested by the following results.

Theorem 4.3 *Let h_1, h_2 be prefixing functions, where $n^\epsilon \leq h_1(n) \leq h_2(n)$ for sufficiently large n, for some $\epsilon > 0$. Suppose also that $n - h_1(n)$ is nondecreasing. Then if all languages in P have h_1-prefix-PTC's (PTD's), then all languages in P have h_2-prefix-PTC's (PTD's).*

Proof: Let $p(n) = n^k$ where $k \geq 1/\epsilon$. Let $g_1(n) = n - h_1(n)$ and $g_2(n) = n - h_2(n)$. We have $h_1(p(n)) - h_2(n) = p(n) - g_1(p(n)) - n + g_2(n) \leq p(n) - n$ since $g_2(n) \leq g_1(n) \leq g_1(p(n))$. Also $h_1(p(n)) - h_2(n) = h_1(n^k) - h_2(n) \geq n - h_2(n) \geq 0$. The result follows by Lemma 3.6. □

Again, if $h_2(n) \leq h_1(n)$, then the existence of h_1-prefix-PTC's implies the existence of h_2-prefix-PTC's if h_1 and h_2 are not too far apart, as measured this time by $n - h_1(n)$ and $n - h_2(n)$.

Theorem 4.4 *Let h_1, h_2 be prefixing functions, h_1 is nondecreasing, $h_2(n) \leq h_1(n)$ for sufficiently large n. Let $g_1(n) = n - h_1(n)$ and $g_2(n) = n - h_2(n)$. Suppose there is a polynomial p such that $g_1(p(n)) \geq g_2(n)$ for sufficiently large n. Then if all languages in P have h_1-prefix-PTC's (PTD's), then they all have h_2-prefix-PTC's (PTD's).*

Proof: The result follows by a straightforward application of Lemma 3.6. □

Our next theorem provides some more evidence for the conjecture of decreasing difficulty of construction for languages in P as the prefixing function increases from $n - n^\epsilon$ to n. We first need some definitions.

Theorem 3.17 *Let h be a nondecreasing prefixing function, $h(n) \geq \log n$. Let $t : N \to N$ be a function computable in time polynomial in $t(n)$, and p a polynomial such that $n \leq t(h(n)) \leq p(n)$ for sufficiently large n. Assume also that for each constant k, $t(kn) \leq t(n)^k$. Then if*

$$\bigcup_{c>0} NTIME(t(n)^c) = \bigcup_{c>0} TIME(t(n)^c)$$

then all NP languages have h-prefix-PTD's.

Proof: Let L be a language in NP, and define

$$L' = \{x1^{n-h(n)} || |x| = h(n), \exists y \text{ such that } |xy| = n, xy \in L\}$$

To show that L has an h-prefix-PTD it suffices to show that L' is in P. Define

$$L_1 = \{n\#x || |x| = h(n), x1^{n-h(n)} \in L'\}$$

If $|x| = h(n)$ and $m = |n\#x|$, determining whether $n\#x \in L_1$ can be done in nondeterministic time polynomial in n. But $n \leq t(h(n)) \leq t(m)$ and hence by hypothesie L_1 is in $DTIME(t(n)^c)$ for some c. To determine whether $x1^{n-h(n)} \in L'$, where $|x| = h(n)$, it suffices to ask whether $n\#x \in L_1$. Since $|n\#x| \leq kh(n)$ for some constant k, this can be done in deterministic time $t(kh(n))^c \leq t(h(n))^{kc} \leq p(n)^{kc}$. □

An interesting open question is whether a stronger converse of Theorem 3.14 holds, namely whether the fact that NP languages can be recognized in time polynomial in $t(n)$ implies that all NP languages must have h-prefix-PTD's. A relativization result from [2] (see Theorem 9) seems to indicate the contrary.

We have examined the complexity of prefix-based detection and construction for NP languages. Briefly, if $h(n) \geq n^\epsilon$ for some $\epsilon > 0$, then there are NP languages which do not have h-prefix-PTC's, unless P = NP. For smaller h, there are relativizations under which all NP languages do have such PTC's, and relativizations under which they do not. In addition, the last few theorems show what appears to be increasing difficulty of construction as h increases within this range. These remarks also apply to prefix-based PTD's. An open question is whether the existence of h-prefix-PTD's implies the existence of h-prefix-PTC's for NP languages, for the smaller values of h.

4 Construction for languages in P

We now consider languages in P. For small enough functions h, the existence of h-prefix-PTC's follows the same pattern for P and for NP languages, as shown by the following.

Theorem 4.1 *Let h be a nondecreasing prefixing function, and assume that there exists $\epsilon > 0$ such that $h(n) \leq n - n^\epsilon$ for sufficiently large n. Then all languages in P have h-prefix-PTC's (PTD's), if and only if all languages in NP have h-prefix-PTC's (PTD's).*

We have that S'' is in NP, and using a similar argument to that used in [2], it can be seen that if S'' is in P, then S is in P. But if S'' has an h-prefix-PTD, then S'' must be in P (note $h(t) \geq h(n)$). □

It is interesting to consider some special cases of the above theorem. When $h(n) = 0$ the theorem states in particular that if all languages in NP have 0-prefix-PTD's then there are no tally languages in NP - P. By considering all functions h for which $h(n) \leq k \log n$ for some k, the theorem states that all NP languages have h-prefix-PTD's for all such functions h, if and only if there are no sparse languages in NP - P. Since it has been shown that there are no tally languages in NP - P if and only if there are no sparse languages in NP - P (see [2]), this provides an alternate proof for one direction of Corollary 3.9. When $h(n) = n$, the theorem states the obvious fact that all NP languages have h-prefix-PTD's if and only if P=NP. We also have the following corollary by Theorem 3.2.

Corollary 3.13 *Let $h(n)$ be a prefixing function such that $h(n) \geq n^{\epsilon}$ where $0 < \epsilon < 1$, for sufficiently large n. Let $\delta(n) = 2^{h(n)}$. Then there are no $\delta(n)$ uniformly distributed dense sets in NP – P if and only if P = NP.*

Because of the preceding lemma, the following theorem can be seen as a generalization of part of Theorem 8 in [2]. The proof is similar to that of Theorem 8 and is omitted.

Theorem 3.14 *Let h be a prefixing function, and let $t : N \rightarrow N$ be such that $h(t(n)) \geq n$ for sufficiently large n. Suppose also that $t(n)$ is computable in time polynomial in $t(n)$. Then if all NP languages have h-prefix-PTD's, then*

$$\bigcup_{c>0} NTIME(t(n)^c) = \bigcup_{c>0} TIME(t(n)^c)$$

and in this case each language in NP is recognizable in time $q(t(n))$ for some polynomial q.

Note that Theorem 3.2 is actually a corollary of this theorem. It is useful to list a couple of other corollaries to see the implications of the existence of h-prefix-PTD's for varying h.

Corollary 3.15 *Let $h(n) = \lceil (\log n)^r \rceil$ for some integer r. Then if all languages in NP have h-prefix-PTD's, then each language in NP can be recognized in time $2^{kn^{1/r}}$ for some constant k.*

Corollary 3.16 *Let $h(n) = \lceil 2^{(\log n)^{1/r}} \rceil$ for some integer r. Then if all languages in NP have h-prefix-PTD's, then each language in NP can be recognized in time $2^{k(\log n)^r}$ for some constant k.*

The above theorem has a partial converse, again a generalization of part of Theorem 8 in [2].

a string y such that $|xy| = m$ and such that xy was not queried in PO_i's computation (this can be done since $2^{m-h_2(m)} > p_i(m)$). Put the string xy in B. □

We now relate the existence of h-prefix-PTD's and PTC's to the existence of subexponential algorithms for NP languages. This will provide another demonstration of the apparent increase in the difficulty of h-prefix construction as h increases from 0 to n^ϵ. The proof also involves a generalization of a result given in [5], which states that all languages in NP have 0-prefix-PTD's if and only if there are no sparse languages in NP-P.

We will make use of results derived by Hartmanis, Immerman, and Sewelson in [2]. The following definitions are similar to those from [2]. (In computing density, we count the strings of length *exactly* n, rather than of length *at most* n, as is done in [2])

Definition 3.1 A set A has density $\delta(n)$ if for each n, the number of strings in A of length n is at most $\delta(n)$. A language is sparse if it has polynomial density.

Definition 3.2 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 of size n, where an interval is any set of strings consecutive in the lexicographic ordering of Σ^*.

The following lemma relates these definitions to the existence of PTD's.

Lemma 3.12 *Let $h(n)$ be a nondecreasing prefixing function, and let $\delta(n) = 2^{h(n)}$. Then all languages in NP have h-prefix-PTD's if and only if there are no $\delta(n)$ uniformly distributed dense sets in NP – P.*

Proof: Let L be any language in NP. Define

$$L' = \{x1^{n-h(n)} \mid |x| = h(n) \text{ and there exists } y, |y| = n - h(n) \text{ such that } xy \in L\}$$

Note that L' is a $\delta(n)$ uniformly dense set in NP, and that L has an h-prefix-PTD if and only if L' is in P. Thus if there are no $\delta(n)$ uniformly distributed dense sets in NP - P, then L has an h-prefix-PTD.

The proof of the converse uses the technique used in [2] to show that there are no sparse languages in NP-P if NEXPTIME = EXPTIME. Suppose that all NP languages have h-prefix-PTD's, and let S be any $\delta(n)$ uniformly distributed dense set in NP. Define

$$\begin{aligned} S' = \{&z\#n\#i\#j\#k\#d \mid \exists x_1 < x_2 < \ldots < x_i \leq x < y_1 < y_2 < \ldots y_j \in S \\ &|x_1| = |y_j| = n, |z| = h(n), z \sqsubset x_1, x_2, \ldots, x_i, x, y_1, \ldots y_j, \text{ and the } k\text{th bit of } x \text{ is } d\} \end{aligned}$$

Note that if $z\#n\#i\#j\#k\#d \in S'$, i and j are bounded by a polynomial in n, so there exists a constant r such that for all n, if $z\#n\#i\#j\#k\#d \in S'$, then the length of the encoding $n\#i\#j\#k\#d$ is at most $r \log n$. We can also assume without loss of generality that the length of this encoding is at least $\log(n+1)$. Define

$$S'' = \{z01^{t-h(n)-1} \mid t \text{ is the binary number encoded by the string } n\#i\#j\#k\#d, \text{ where } z\#n\#i\#j\#k\#d \in S'\}$$

Corollary 3.9 *Let h be a prefixing function such that $h(n) \leq k \log n$ for some constant k for sufficiently large n. Then all languages in P (NP) have 0-prefix-PTC's (PTD's) if and only if they all have h-prefix-PTC's (PTD's).*

Although we have stated and proved Theorems 3.7 and 3.8 for $h(n) \leq n - n^{\epsilon}$, for $h(n) \geq n^{\epsilon'}$ the results already follow by Theorem 3.2. For smaller h, Theorem 3.8 is in a sense the best that can be shown using techniques that relativize. For by extending the relativization from Theorem 3.5 we can show that when h_1 and h_2 are further apart than specified in Theorem 3.8, then an oracle exists relative to which all NP languages have h_1-prefix-PTC's but not all P languages have h_2-prefix-PTC's. Again we need the following lemma.

Lemma 3.10 *Let h_1, h_2 be nondecreasing prefixing functions such that for all polynomials p and constants k, $h_2(n) - h_1(p(n)) > k \log n$ for infinitely many n. Then for all polynomial p and constants k, the inequality holds for infinitely many even n.*

Proof: Given polynomial p and constant k, let r be such that $p(n) \leq n^r$. Define $q(n) = n^{4r}$ and $k' = 4k$. By hypothesis, there exists an infinite increasing sequence $m_1, m_2, \ldots$ such that $h_1(q(m_i)) < h_2(m_i) - k' \log m_i$. For each i define $n_i = m_i(m_i + 1)$. Note that $p(n_i) \leq (m_i^2 + m_i)^r \leq m_i^{4r} = q(m_i)$ for sufficiently large n_i. Also $k \log n_i = k \log (m_i^2 + m_i) \leq 2k \log (m_i + 1) \leq 4k \log m_i = k' \log m_i$. Thus we have for sufficiently large n_i,

$$h_1(p(n_i)) \leq h_1(q(m_i)) < h_2(m_i) - k' \log m_i \leq h_2(n_i) - k \log n_i$$

□

Theorem 3.11 *Let h_1, h_2 be nondecreasing prefixing functions such that $h_1(n) \leq h_2(n)$. Suppose that for all polynomials p and constants k, $h_2(n) - h_1(p(n)) > k \log n$ for infinitely many n, and that $h_2(n) < n - k \log n$ for sufficiently large n. Then there exists an oracle B such that all languages in NP^B have h_1-prefix-PTC's relative to B, but not all languages in P^B have h_2-prefix-PTD's relative to B.*

Proof: The construction is similar to that of Theorem 3.5. Strings are coded into the oracle in exactly the same manner to ensure that all languages in NP^B have h_1-prefix-PTC's relative to B. Instead of constructing a language in $\mathrm{NP}^B - \mathrm{P}^B$, however, we manipulate the oracle so that the language B cannot have an h_2-prefix-PTD.

At stage i the construction ensures that the machine PO_i is not an h_2-prefix-PTD for B. Let $q(n) = p_i^3(n)$, and let k be an integer such that $\log q(n) \leq k \log n$. By hypothesis, and the preceding lemma, it is possible to choose even m such that $h_2(m) - h_1(p_i(m)) \geq k \log m \geq \log q(m)$, $h_2(m) < m - k \log m$, and so that no strings of length greater than or equal to m have been put in B or queried in previous stages. We therefore have $2^{h_2(m)} > p_i^3(m) 2^{h_1(p_i(m))}$ and $2^{m - h_2(m)} > p_i(m)$. After coding in the strings of odd length less than or equal to $p_i(m)$, as described in the proof of Theorem 3.5, choose a string x of length $h_2(m)$ such that no string of length m with prefix x has been fixed to be in or out of B. This can be done by the choice of m. Run PO_i on input $(1^m, x)$. If PO_i outputs 1 do nothing. Otherwise, choose

Proof: Choose integer $k \geq 1/\epsilon$ and apply Lemma 3.6 with $p(n) = n^k$. □

If h_1 and h_2 are not too far apart, then the same type of result follows also when $h_1(n) \leq h_2(n)$.

Theorem 3.8 *Suppose that h_1, h_2 are prefixing functions, h_1 and h_2 are nondecreasing, $h_1(n) \leq h_2(n) \leq n - n^\epsilon$ for some ϵ, where $0 < \epsilon < 1$, for sufficiently large n. Suppose also that there exists an integer k and a polynomial p such that for sufficiently large n, $h_2(n) - h_1(p(n)) \leq k \log n$. Then if all languages in P (NP) have h_1-prefix-PTC's (PTD's), then all languages in P (NP) have h_2-prefix-PTC's (PTD's).*

Proof: Let $\rho : N \times N \to N$ be a polynomial-time computable bijection between $N \times N$ and N. We may assume that $\rho(n, m) \geq n$ for all n and that there is a polynomial q such that $\rho(n, m) \leq q(max(n, m))$ for all n, m. For each string y, let $\beta(y)$ denote the binary integer encoded by y. Let $k_1 > k$ be such that $k \log n \leq k_1 \lfloor \log n \rfloor$.

Define $h_2'(n) = max(h_2(n), k_1 \lfloor \log n \rfloor)$. Note that h_2' satisfies all the assumptions made concerning h_2 in the statement of the theorem, with $k_1 \lfloor \log n \rfloor$ in place of $k \log n$. We will show that if all P (NP) languages have h_1-prefix-PTC's, then all P (NP) languages have h_2'-prefix-PTC's. Since $h_2(n) \leq h_2'(n)$, it will follow by the previous theorem that all P (NP) languages have h_2-prefix-PTC's as well.

For each string x of length n, where $x = x_1 x_2$ and $x_1 \sqsubset_{k_1 \lfloor \log n \rfloor} x$, let $s(x)$ be the string of length $\rho(n, \beta(x_1))$ which contains the string x_2 padded with additional 0's. Note that since $\beta(x_1) \leq n^{k_1}$, we have $|s(x)| \leq q(n^{k_1})$. If L is a language in P (or NP), define $L' = \{s(x) | x \in L\}$. Clearly L' is also in P (NP).

Let $g_1(n) = n - h_1(n)$, and $g_2(n) = n - h_2'(n)$. Since $g_1(n) \geq n^\epsilon$, for large enough polynomials r, $g_1(r(n)) \geq q(n^{k_1}) + n$. Also we have that $h_1(p(n)) \geq h_2'(n) - k_1 \lfloor \log n \rfloor$. Since h_1 is nondecreasing, we have $h_1(r(n)) \geq h_2'(n) - k_1 \lfloor \log n \rfloor$ for large enough polynomials r as well. Choose strictly increasing polynomial r satisfying both inequalities.

Now let L'' be the language consisting of all strings of the form $x_1 0^{m_1} x_2 0^{m_2}$ of length $r(\rho(n, m))$, where $x = x_1 x_2 \in L'$ has length $\rho(n, m)$ for some $m \leq n^{k_1}$, $x_1 \sqsubset_{h_2'(n) - k_1 \lfloor \log n \rfloor} x$,

$$m_1 = h_1(r(\rho(n, m))) - h_2'(n) + k_1 \lfloor \log n \rfloor, \text{ and}$$

$$m_2 = r(\rho(n, m)) - \rho(n, m) - h_1(r(\rho(n, m))) + h_2'(n) - k_1 \lfloor \log n \rfloor.$$

We have $m_1 \geq 0$ by hypothesis and choice of r. Also

$$m_2 = r(\rho(n, m)) - \rho(n, m) - r(\rho(n, m)) + g_1(r(\rho(n, m))) + n - g_2(n) - k_1 \lfloor \log n \rfloor \geq 0$$

since $g_1(r(\rho(n, m))) \geq q(n^{k_1}) + n \geq \rho(n, m) + g_2(n)$. It is not hard to see that L'' is in P (NP), and that if L'' has an h_1-prefix-PTC (PTD), then L has an h_2'-prefix-PTC (PTD). □

We have the following corollary, showing that the existence of length-based PTC's is enough to guarantee the existence of h-prefix-PTC's for prefixing functions h which are $O(\log n)$.

to be out of B). Choose the smallest such string x and put the string $0^j 1x10^{p_j(k)}10^n$ and all of its prefixes into B. (In this context, the prefixes of the string $w10^n$ (e.g. $w = 0^j 1x10^{p_j(k)}$) are all strings of the form $y10^{2n-m}$ where y is a prefix of w of length m). Also choose one accepting computation of N_j on x and put the strings of length m which must be added to B to make the computation accepting into B. Fix any other strings queried during this computation, and getting a negative response, to be in $\overline{B}$.

After processing all n such that $2n + 1 \leq p_i(m)$, no more than $p_i^3(m)2^{h(p_i(m))}$ strings of length m will have been fixed to be in or out of B. By choice of m, this quantity is less than $2^{m/2}$. So there is at least one string w of length $m/2$ such that no string wy of length m has been fixed to be in or out of B. Run P_i on string string w using as oracle the strings put into B so far. If P_i accepts do nothing. Otherwise choose some string y of length $m/2$ such that wy was not queried in P_i's computation on w (this can be done since $p_i(m/2) < 2^{m/2}$) and add wy to B. □

By the previous results we see that for larger prefixing functions h, not all NP languages can have h-prefix-PTC's unless P=NP. For smaller h, this is an open question, as indicated by the two contrasting relativizations. Even within the range delimited by these smaller prefixing functions, however, there appears to be a pattern of increasing difficulty for efficient construction, as h increases. This will be demonstrated by the following results. First we show that if $h_1(n) \geq h_2(n)$ and both h_1 and h_2 are not too large, then the existence of h_1-prefix-PTC's implies the existence of h_2-prefix-PTC's. Then we present a relativization under which if h_1 and h_2 are far enough apart, all NP languages have h_2-prefix-PTC's but not all languages in P have h_1-prefix-PTC's.

The following lemma will be used to prove several results concerning the relationship between the existence of h_1 and h_2-prefix-PTC's or PTD's.

Lemma 3.6 *Let h_1, h_2 be prefixing functions, and suppose that there exists a strictly increasing polynomial p such that for sufficiently large n, $p(n) - n \geq h_1(p(n)) - h_2(n) \geq 0$. Then if all languages in P (NP) have h_1-prefix-PTC's (PTD's), then all languages in P (NP) have h_2-prefix-PTC's (PTD's).*

Proof: Let L be any language in P (or NP). Define

$$\begin{aligned} L' \;=\; & \{x_1 0^{m_1} x_2 0^{m_2} | x = x_1 x_2 \in L, |x| = n, x_1 \sqsubset_{h_2(n)} x, \\ & m_1 = h_1(p(n)) - h_2(n), \text{ and } m_2 = p(n) - h_1(p(n)) - n + h_2(n)\} \end{aligned}$$

The hypothesis implies that $m_1, m_2 \geq 0$, and clearly L' is in P (or NP). It is not hard to see that if L' has an h_1-prefix-PTC (PTD), then L has an h_2-prefix-PTC (PTD). □

Theorem 3.7 *Suppose that h_1, h_2 are prefixing functions, h_1 is nondecreasing, and $h_2(n) \leq h_1(n) \leq n - n^\epsilon$ for some ϵ, where $0 < \epsilon < 1$, for sufficiently large n. Then if all languages in P (NP) have h_1-prefix-PTC's (PTD's), then all languages in P (NP) have h_2-prefix-PTC's (PTD's).*

Theorem 3.3 *Let h be a prefixing function such that for each constant k, $h(n) < n - k\log n$ for infinitely many n. Then there exists an oracle A and a language $L \in P^A$ such that L does not have an h-prefix-PTD relative to A.*

Proof: This follows by a simple diagonalization construction (see for example [1]) which ensures that $A \in \mathrm{P}^A$ does not have an h-prefix-PTD. To ensure that PO_i is not an h-prefix-PTD for A, choose k such that $p_i(n) \leq n^k$ and choose n such that $h(n) < n - k\log n$. Run PO_i on input $(1^n, 0^{h(n)})$. If PO_i outputs 1, do nothing. Otherwise choose a string of length n with prefix $0^{h(n)}$ not queried by PO_i's computation and put it in A. This can be done since $2^{n-h(n)} > n^k$. □

For the opposite relativization we need the following simple lemma.

Lemma 3.4 *Let h be a nondecreasing prefixing function such that for each $\epsilon > 0$, $h(n) < n^\epsilon$ for infinitely many n. Then for all positive integers k, and all polynomials q, $h(n^k) < n/2 - \log q(n)$ for infinitely many even n.*

Proof: Let $\epsilon = 1/(2k)$. By hypothesis there exists an infinite increasing sequence of integers $m_1, m_2, \ldots$ such that $h(m_i) < m_i^\epsilon$. For each i, let n_i be the largest even integer such that $n_i^k \leq m_i$. Thus $m_i < (n_i + 2)^k$. We have

$$h(n_i^k) \leq h(m_i) < m_i^\epsilon < (n_i + 2)^{k\epsilon} = (n_i + 2)^{1/2} < n_i/2 - \log q(n_i)$$

for sufficiently large n_i. □

Theorem 3.5 *Let h be a nondecreasing prefixing function such that for each $\epsilon > 0$, $h(n) < n^\epsilon$ for infinitely many n. Then there exists an oracle B such that $P^B \neq NP^B$ and such that all languages in NP^B have h-prefix-PTC's relative to B.*

Proof: Two types of strings are put into B at each stage of the construction. Strings of even length are used to ensure that the language $L_B = \{x | \exists y \text{ such that } |x| = |y| \text{ and } xy \in B\}$ is in $\mathrm{NP}^B - \mathrm{P}^B$. Strings of odd length are used to ensure that all languages in NP^B have h-prefix-PTC's relative to B.

At stage i the construction ensures that the machine P_i does not accept L_B. Let r be an integer such that $p_i(n) \leq n^r$. By the previous lemma, we can choose an even integer m such that $2^{m/2} > p_i^3(m)2^{h(m^r)}$, and such that no strings of length greater than or equal to m have been added to B or have been queried in computations in previous stages.

Before deciding membership in B for strings of length m such membership is decided for strings of length $2n+1$ where $2n+1 \leq p_i(m)$. At the same time some strings of length m are reserved for $\overline{B}$, in other words these strings are "fixed" to be out of B. For each n such that no processing has been done yet for strings of length $2n+1$ and such that $2n+1 \leq p_i(m)$, in order from smallest such n to largest such n, the following is done. Determine all pairs j, k such that $j + 1 + k + 1 + p_j(k) = n$. For each such pair, and for each string z of length $h(k)$, determine which strings of length k with prefix z (if any) would be accepted by N_j (with oracle B) if other strings of length m were added to B (other than those already fixed

3 Construction for languages in NP

Clearly the existence of h-prefix-PTC's implies that of h-prefix-PTD's. Most of the following results apply to both PTC's and PTD's, and when possible are stated in the form which gives the stronger statement. Because of the following theorem, it suffices to examine the complexity of construction and detection under the assumption that P $\neq$ NP.

Theorem 3.1 *If P=NP, then all languages in NP have h-prefix-PTC's for any prefixing function h.*

Proof: If L is in NP, define

$$L' = \{xy || x| = |y| = n \text{ and } \exists w \in L \text{ such that } x \leq w \leq y\}$$

Since clearly L' is in NP, it is by hypothesis in P. Given $(1^n, z)$ where $|z| = h(n)$, a binary search on L' suffices to find a string of length n in L with prefix z. □

The following provides a partial converse to the above result, applying to NP languages and large enough functions h.

Theorem 3.2 *Let h be a prefixing function such that $h(n) \geq n^\epsilon$ for some ϵ, $0 < \epsilon \leq 1$, for sufficiently large n. The following are equivalent:*

1. *P = NP.*
2. *All languages in NP have h-prefix-PTC's.*
3. *All languages in NP have h-prefix-PTD's.*

Proof: By the previous result, it suffices to show that (3) implies (1). Let SAT be the language consisting of suitable encodings of satisfiable formulas (an NP-complete language). Let k be an integer such that $k \geq 1/\epsilon$. Note that $h(n^k) \geq n$. Define

$$S = \{xy || x| = n, |xy| = n^k, x \in SAT\}$$

S is in NP, and the existence of an h-prefix-PTD M for S implies that SAT is in P, as follows. To determine whether $x \in SAT$, where $|x| = n$, it suffices to run M on input $(1^{n^k}, x0^{h(n^k)-n})$.

□

This result contrasts with those obtained in [5], showing that the existence of 0-prefix-PTC's for languages in NP cannot be determined using techniques that relativize, even under the assumption that P $\neq$ NP. This is in fact the case for prefixing functions which are smaller than the ones considered in the last theorem. For these values, assuming that P $\neq$ NP, there exist relativization results for both of the possible conclusions, namely that all NP (or P) languages have h-prefix-PTC's, or that they don't. The simpler relativization is the latter. (It actually applies to a larger range of prefixing functions).

(i.e. construction based on the specification of given parameter values for the string) can in some cases be derived from this type of "prefix" construction as well.

Definition 1.2 A function $h : N \rightarrow N$ is called a prefixing function if $0 \leq h(n) \leq n$ for all $n \in N$, and h is computable in polynomial time.

Definition 1.3 Let $h : N \rightarrow N$ be a prefixing function.

1. An h-prefix-PTD for a language L is a polynomial-time machine which given as input $(1^n, z)$, where $|z| = h(n)$, outputs 1 if there exists a string in L of length n having prefix z, and outputs 0 otherwise.

2. An h-prefix-PTC for a language L is a polynomial-time machine which given as input $(1^n, z)$, where $|z| = h(n)$, outputs a string $x \in L$ of length n having prefix z, if such a string exists, and outputs Λ otherwise.

The case where $h(n) = 0$ gives us regular PTC's and PTD's and was dealt with in [5]. The other extreme, $h(n) = n$, gives us the recognition problem and is therefore easy for languages in P, whereas its complexity for NP languages depends on the P=NP question. As previously mentioned, there is an interesting pattern for both P and NP languages as h increases from 0 to n. This is investigated in the following sections.

2 Notation

We consider all languages to consist of strings over the alphabet $\{0, 1\}$. We will write $x \sqsubset_n y$ to signify that x is a prefix of y of length n. More generally, $x \sqsubset y$ means that x is a prefix of y. The notation $x \approx_n y$ means that x and y share a prefix of length n (i.e. they are identical in the first n bits). We assume the usual lexicographic ordering on strings.

For the relativization results, we let $P_1, P_2, ...$ be an effective enumeration of the deterministic oracle Turing machine acceptors that run in polynomial time; we let $N_1, N_2, ...$ be an effective enumeration of the nondeterministic oracle Turing machine acceptors that run in polynomial time; and we let $PO_1, PO_2, ...$ be an effective enumeration of the deterministic oracle Turing machine transducers (machines that produce an output) that run in polynomial time. Without loss of generality we let $p_1, p_2, ...$ be strictly increasing polynomials such that p_j bounds the running times of P_j, N_j, and PO_j.

We say that a language L has a PTC (or PTD) relative to an oracle A if there exists an oracle Turing machine which when querying oracle A acts like a PTC (or PTD) for L.

If $f : N \rightarrow N$ is a function, we denote by $TIME(f(n))$ the class of languages that can be recognized in time $f(n)$, and by $NTIME(f(n))$ the class of languages recognizable by a nondeterministic machine in time $f(n)$.

1 Introduction

The recognition problem for a language L consists of determining, for a given element x, whether x belongs to L. The fact that a language L is in P, means of course that the recognition problem for L can be solved in polynomial time. Thus, for example, the language consisting of all CNF formulas which are satisfiable is in P if and only if P=NP.

A different type of problem, which can arise in some applications, consists of producing an element from a given language having some given characteristics. For example, one may wish to produce a satisfiable CNF formula of a certain size, or a graph that can be colored with k colors. Many types of properties can be associated with the desired instance; the most obvious of these is the size.

The complexity of construction based on length or size was investigated in [5] for languages in P and NP. The following definitions are from [5].

Definition 1.1

1. A polynomial-time detector (PTD) for a language L is a polynomial-time machine which given as input 1^n outputs 1 if there exists a string in L of length n, and outputs 0 otherwise.

2. A polynomial-time constructor (PTC) for a language L is a polynomial-time machine which given as input 1^n outputs a string $x \in L$ of length n, if such a string exists, and outputs Λ otherwise.

It was shown in [5] that although most well-known languages in P and NP appear to have PTC's as well as PTD's, the question of whether or not all P or NP languages have such machines cannot be resolved using techniques that relativize, even under the assumption that P $\neq$ NP. Thus the apparent difficulty of the recognition problem for NP-complete languages does not necessarily imply the same difficulty for construction and detection of such languages; whereas for languages in P, construction and detection might actually be harder than recognition.

The results just mentioned are actually a special case of more general results about construction and detection for languages in P and NP. We consider here a more general class of construction and detection problems, involving the specification of additional information concerning the desired string. We show that an interesting structure exists for these problems based on the amount of information that is specified about the string to be produced. In very general terms, our results show that for languages in NP, the difficulty of construction and detection appears to increase as more information is provided about the desired element. For languages in P, however, the difficulty of construction increases and then again decreases as the amount of specified element information is increased.

One conceptually simple way to provide a measurable amount of additional information about the desired string, and thus to narrow the set of elegible strings which can be produced, is to specify a prefix for the string, as well as its length. This is the approach taken in this paper. It will be shown in a later section that more general parameter-based construction

ABSTRACT

We investigate the problem of when it is possible to efficiently construct an instance of a given language in P or NP, based on partial information provided about the desired instance. We model this problem by specifying a prefix and a length for the string to be produced. Our results suggest that it may be harder to find efficient constructors for languages in NP if more information is provided about the desired element (assuming that P is not equal to NP). Specifically, for large enough prefix functions, polynomial-time prefix-based constructors cannot exist for all languages in NP unless P=NP. For smaller prefix functions, the existence of such constructors cannot be determined using techniques that relativize, under the assumption that P $\neq$ NP. We also show through relativizations and other results, what appears to be a pattern of increasing difficulty for efficient construction as the prefix function increases within this range. For languages in P, however, the difficulty of construction first increases and then appears to decrease as the size of the prefix is increased. We also relate prefix-based construction to that based on more general parameter functions for the strings. In addition, we examine the related ranking problem in the same context, and prove some results about the existence of prefix-based rankers for languages in P and NP.

DIMACS Technical Report 93-31
May 1993

Constructing Language Instances Based on Partial Information

by

Laura A. Sanchis[1,2]
Department of Computer Science
Colgate University
Hamilton, NY 13346

[1]Supported in part by DIMACS.
[2]Supported in part by NSF grant CCR-9101974.

DIMACS is a cooperative project of Rutgers University, Princeton University, AT&T Bell Laboratories and Bellcore.

DIMACS is an NSF Science and Technology Center, funded under contract STC–91–19999; and also receives support from the New Jersey Commission on Science and Technology.