Quasilinear Time Complexity Theory

Ashish V. Naik\textsuperscript{1}  
SUNY Buffalo  
avnaik@cs.buffalo.edu

Kenneth W. Regan\textsuperscript{2}  
SUNY Buffalo  
regan@cs.buffalo.edu

D. Sivakumar  
SUNY Buffalo  
sivak-d@cs.buffalo.edu

\textsuperscript{1}Supported in part by NSF grant CCR-9002292.

\textsuperscript{2}Supported in part by NSF grant CCR-9011248.
Abstract

This paper furthers the study of quasi-linear time complexity initiated by Schnorr [Sch76] and Gurevich and Shelah [GS89]. We show that the fundamental properties of the polynomial-time hierarchy carry over to the quasilinear-time hierarchy. Whereas all previously known versions of the Valiant-Vazirani reduction from NP to parity run in quadratic time, we give a new construction using error-correcting codes that runs in quasilinear time. We show, however, that the important equivalence between search problems and decision problems in polynomial time is unlikely to carry over: if search reduces to decision for SAT in quasi-linear time, then all of NP is contained in quasi-polynomial time. Other connections are made to work by Stearns and Hunt [SH86, SH90, HS90] on “power indices” of NP languages, and to work on bounded-query Turing reductions and helping by robust oracle machines.
1 Introduction

The notion of "feasible" computation has most often been identified with the concept of polynomial time. However, an algorithm that runs in time $n^{100}$ or even time $n^2$ may not really be feasible on moderately large instances. Quasi-linear time, namely time $qlin := n \cdot (\log n)^{O(1)}$, reduces the problem of the exponent of $n$. Let DQL and NQL stand for time $qlin$ on deterministic and nondeterministic Turing machines. Schnorr [Sch76, Sch78] showed that SAT is complete for NQL under DQL many-one reductions ($\leq_{m}^{e}$). Together with Stearns and Hunt [SH86, SH90], it was shown that many known NP-complete problems also belong to NQL and are complete for NQL under $\leq_{m}^{e}$, so that the NQL vs. DQL question takes on much the same shape as NP vs. P. Related classes within P are studied by Buss and Goldsmith [BG93].

One theoretical difficulty with the concept of quasi-linear time is that it appears not to share the degree of independence on particular machine models that makes polynomial time such a robust concept. Gurevich and Shelah [GS89] showed that a wide variety of models related to the RAM under log-cost criterion [CR73] accept the same class of languages in quasi-linear time—we call this class DNLT. They also showed that nondeterministic $qlin$ time for these machines, namely NNLT, equals NQL. However, currently it appears that DNLT is larger than DQL, and that for all $d > 1$, Turing machines with $d$-dimensional tapes accept more languages in time $qlin$ than do TMs with $(d - 1)$-dimensional tapes (cf. [WW86]). Our constructions all work for DQL as well as DNLT.

Our main motivation is to ask: How much of the known theory of complexity classes based on polynomial time carries over to the case of quasi-linear time? Section 2 observes that the basic results for the polynomial hierarchy hold also for the quasi-linear hierarchy.

Section 3 shows that the randomized reduction from NP to parity given by Valiant and Vazirani [VV86] and used by Toda [Tod91], which was previously proved by constructions that run in quadratic time (see [VV86, Tod91, CRS93, Gup93]), can be made to run in quasi-linear time. Our construction also markedly improves both the number of random bits needed and the success probability, and uses error-correcting codes in an interesting manner first noted in [NN90].

Section 4 studies what may be the major difference between polynomial and quasi-linear time: the equivalence between functions and sets seems no longer to hold. It has long been known that every function can be computed in polynomial time using some set as an oracle. In contrast, we show that there exist functions that cannot be computed in quasi-linear time using any set as an oracle whatsoever. Many natural problems in NP have associated search functions that reduce to the decision problems in polynomial time—in most cases, quadratic time (cf. [Sel88, JY90]). Theorem 4.2 shows that for
SAT, search does not reduce to decision in quasilinear time, unless all of NP is contained in quasi-polynomial time, viz. DTIME[2polylog n]. We also show that the quadratic upper bound is tight unless the power index of SAT is less than 1, which would be contrary to a conjecture of Stearns and Hunt [SH90].

Section 5 shows how our notion of counting the number of query bits used by oracle machines relates to previous work on counting queries [Bei87b, AG88, BGH89, ABG90, Bei91, BGGO93, HN93, BKS93] and on “helping” [Sch85, Ko87, Bal90]. We observe that the known equivalence between having search reduce to decision and one-sided helping in polynomial time carries over to any reasonable time bound t(n). This yields other forms of our main results in Section 4. We construct an oracle A relative to which search reduces to decision for SAT in quasilinear time (in fact, O(n log^2 n) time), but still NP^A \neq P^A, so that SAT relative to A is still “P^A-superease” (see [BKS93]). This also gives evidence that our quasipolynomial simulation of NP in Theorem 4.2 is close to optimal. A concluding Section 6 summarizes the significance of this work and suggests some problems for further research.

2 Notation and Basic Results

Let \( \Sigma := \{0, 1\} \). Given strings \( y_1, \ldots, y_m \in \Sigma^* \), each \( y_i \) of length \( n_i \), let \( y = (y_1, \ldots, y_m) \) stand for the binary string of length \( 2m + \sum n_i \) obtained by translating 0 to 00, 1 to 11, and ‘comma’ to 01, with an extra 01 at the end. For any language \( R \) we often write \( R(x, y) \) in place of \( \langle x, y \rangle \in R \) and consider \( R \) to be a predicate. For convenience we call \( q \) a quasilinear function if there are constants \( k, c, d \geq 0 \) such that for all \( n \),

\[
q(n) = cn(\log^k n) + d.
\]

Where \( n \) is understood we write \( q \) as short for \( q(n) \), and also write \((\exists y)\) for \((\exists y \in \{0, 1\}^{q(n)})\), \((\forall y)\) for \((\forall y \in \{0, 1\}^{q(n)})\). The notation \((\#_y : R(x, y))\) means “the number of strings \( y \in \{0, 1\}^{q(|x|)} \) such that \( R(x, y) \) holds.”

**Definition 2.1.** If \( A \in NP \), \( R \in P \), and \( p \) is a polynomial such that for all \( x, x \in A \iff (\exists y) R(x, y) \), then we call \( R \) a witness predicate for \( A \), with the length bound \( p \) understood. We use the same terms in the context of NQL and DQL.

We note the following provision about oracle Turing machines \( M \) made standard in both [WW86] and [BDG88] (see also [LL76, Wra77, Wra78]): Whenever \( M \) enters its query state \( q \), with the query string \( z \) on its query tape, \( z \) is erased when the oracle gives its answer. If the oracle is a function \( g \), we suppose that \( g(z) \) replaces \( z \) on the query tape in the next step.

If \( A \) and \( B \) are languages such that \( L(M^B) = A \) and \( M^B \) runs in quasilinear time, then we write \( A \preceq \q{1}{1} B \). As usual we may also write \( A \in DQL(B) \) or \( A \in DQL(B) \), and
if $M$ is nondeterministic, $A \in \text{NQL}^B$ or $A \in \text{NQL}(B)$. Henceforth our notations and definitions of complexity classes are standard, with ‘P’ replaced by ‘QL’, except that we use square brackets for “class operators”:

**Definition 2.2.** For any languages $A$ and $R$,

(a) $A \in \text{NQL}[R]$ if there is a quasilinear function $q$ such that for all $x \in \Sigma^*$, $x \in A \iff (\exists y) R(x, y)$.

(b) $A \in \text{UQL}[R]$ if there is $q$ such that for all $x \in \Sigma^*$, $x \in A \implies (\#^q y : R(x, y)) = 1$, and $x \notin A \implies (\#^q y : R(x, y)) = 0$.

(c) $A \in \oplus \text{QL}[R]$ if there is $q$ such that for all $x, x \in A \iff (\#^q y : R(x, y))$ is odd.

(d) $A \in \text{BQL}[R]$ if there is a quasilinear function $q$ such that for all $x \in \Sigma^*$, $x \in A \implies (\#^q y : R(x, y)) / 2^k > 2/3$, and $x \notin A \implies (\#^q y : R(x, y)) / 2^k < 1/3$.

(e) $A \in \text{RQL}[R]$ if there are $q$ and $\epsilon > 0$ such that for all $x \in \Sigma^*$, $x \in A \implies (\#^q y : R(x, y)) / 2^k > 2/3$, and $x \notin A \implies (\#^q y : R(x, y)) = 0$.

For any class $C$ of languages, $\text{NQL}[C] = \cup_{R \in C} \text{NQL}[R]$, and similarly for the other operators. With $C = \text{DQL}$ these classes are simply written NQL, UQL, $\oplus \text{QL}$, BQL, and RQL. It is easy to check that “machine definitions” of these classes are equivalent to the above “quantifier definitions”; e.g. UQL is the class of languages accepted by unambiguous NTMs that run in quasilinear time. By standard “amplification by repeated trials,” for any function $r = O(\log^k n)$, the classes BQL and RQL remain the same if ‘1/3’ is replaced by $2^{-r(n)}$ and ‘2/3’ by $1 - 2^{-r(n)}$; and similarly for $\text{BQL}[C]$ and $\text{RQL}[C]$ provided $C$ is closed under “polynomial majority truth table reductions.” This is also enough to give $\text{BQL}[\text{BQL}[C]] = \text{BQL}[C]$.

**Definition 2.3.** The *quasilinear time hierarchy* is defined by: $\sum^q_0 = \Pi^q_0 = \Delta^q_0 = \text{DQL}$, and for $k \geq 1$,

$$\sum^q_k = \text{NQL}[\Pi^q_{k-1}], \quad \Pi^q_k = \text{co-} \sum^q_k, \quad \Delta^q_k = \text{DQL} \sum^{q'}_{k-1}.$$  

Also $\text{QLH} := \cup_{k \in \mathbb{N}} \sum^q_k$, and $\text{QLSPACE} := \text{DSPACE}[q \text{lin}]$. By the results of [GS89], all these classes from NQL upward are the same for Turing machines and log-cost RAMs. Next we observe the following concavity property of quasilinear functions. Part (a) is an instance of Jensen’s inequality.

**Lemma 2.1** (a) Let $q(n) = cn \log^k n$, let $n_1, \ldots, n_m$ be nonnegative real numbers, and let $\sum_{i=1}^m n_i \leq r$. Then $\sum_{i=1}^m q(n_i) \leq q(r)$.
(b) If \( q(n) = cn \log^k n + d \), each \( n_i \geq 1 \), and the bound \( r \) in (a) is given by a quasilinear function \( r(n) \), then \( \sum_{i=1}^{m} q(n_i) \) is bounded by a quasilinear function.

**Proof.** (a) True for \( m = 1 \). By the induction hypothesis for \( m - 1 \), \( \sum_{i=1}^{m-1} q(n_i) \leq q(r - n_m) + q(n_m) \). The second derivative of \( q(r - x) + q(x) \) with respect to \( x \) is positive for \( 0 < x < r \), so the maxima on \([0, r] \) are with \( n_m = 0 \) or \( n_m = r \), giving the upper bound \( q(r) \).

(b) By (a), \( \sum_{i=1}^{m} q(n_i) \leq q(r(n)) + dm \). Since each \( n_i \geq 1 \), \( m \leq r(n) \), and so the additive term \( dm \) is quasilinear. If \( r(n) = c'n \log^k n + d' \), then substituting gives a quasilinear bound of the form \( c''n \log^{k+k'} n + d'' \), for some constants \( c'' \) and \( d'' \).

**Corollary 2.2** The relation \( \leq^q \) is transitive. In particular, \( \text{DQL} \supset \text{DQL} \supset \text{DQL} \).

**Proof.** Let \( A = L(M_0^b) \) and \( B = L(M^c) \), where \( M \) runs in time \( q(n) \) and \( M_0 \) in time \( r(n) \). Define \( M_1 \) on any input \( x \) to simulate \( M_0(x) \) but use \( M \) to answer the queries \( y_1, \ldots, y_m \) made by \( M_0 \). For each query \( y_i \), let \( n_i := \max\{ |y_i|, 1 \} \). Then \( \sum_i n_i \) is bounded by \( r(n) \), \( q(n_i) \) bounds the runtime of \( M \) on input \( y_i \), and Lemma 2.1(b) bounds the total runtime of \( M_1 \).

With this in hand it is straightforward to show that the most fundamental properties of the polynomial hierarchy (from [Sto77, Wra77]) carry over to QLH.

**Theorem 2.3** (a) (Equivalence of oracles and quantifiers): For all \( k \geq 1 \), \( \sum_{i=1}^{k} = \text{NQL} \sum_{i=1}^{k} \).

(b) (Downward separation): For all \( k \geq 0 \), if \( \sum_{i=1}^{k} = \Pi_{i=1}^{k} \) then \( \text{QLH} = \sum_{i=1}^{k} \).

(c) (Turing closure): For all \( k \geq 0 \), \( \sum_{i=1}^{k} \cap \Pi_{i=1}^{k} \) is closed downward under \( \leq^q \).

(d) For each \( k \geq 1 \), the language \( B_k \) of quantified Boolean formulas in prenex form with at most \( k \) alternating quantifier blocks beginning with \( \exists \) is complete for \( \sum_{i=1}^{k} \) under DQL many-one reductions.

(e) \( \text{QLH} \subseteq \text{QLSPACE} \).

**Proof.** (a) The base case \( k = 1 \) follows via \( \text{NQL} \supset \text{DQL} = \text{NQL} \supset \text{DQL} \supset \text{NQL} \supset \text{NQL} \) = NQL. The induction case for \( k > 1 \) is typified by showing that \( \text{NQL} \supset \text{Q} \subseteq \sum_{i=1}^{k} \). Let the oracle NTM \( N \) accept \( L \) with oracle \( A \in \text{NQL} \) in quasilinear time \( r(n) \). There is a DQL predicate \( R \) and a quasilinear function \( q \) such that for all \( y \in \Sigma^* \), \( y \in A \iff (\exists v) R(y, v) \). Let \( q(n) = q(r(n)) \) for all \( n \). Then for all \( x \in \Sigma^* \),

\[
x \in L \iff (\exists c)(\exists v)(\forall w) \text{Matrix}(x, c, v, w),
\]
where \( \text{Matrix}(x, \vec{c}, \vec{v}, \vec{w}) \) states that \( \vec{c} \) is an accepting computation of \( N \) on input \( x \) in which some queries \( y_1, \ldots, y_l \) are listed as being answered “yes,” and the other queries \( z_1, \ldots, z_m \) recorded in \( \vec{c} \) are listed as being answered “no,” and \( \vec{v} \) encodes a list of strings \( v_1, \ldots, v_l \) such that \( R(y_1, v_1) \wedge \ldots \wedge R(y_l, v_l) \), and if \( \vec{w} \) encodes a list of strings \( w_1, \ldots, w_m \), then \( \neg R(z_1, w_1) \wedge \ldots \wedge \neg R(z_m, w_m) \). That the quasilinear length bound on the quantification over \( \vec{v} \) and \( \vec{w} \) is sufficient follows from Lemma 2.1(a). Since \( \text{Matrix}(x, \vec{c}, \vec{v}, \vec{w}) \) is decidable in quasilinear time, this is a \( \Sigma^l_2 \) definition of \( L \).

Parts (b) and (c) follow from (a) by standard means. The case \( k = 1 \) of (d) is the main theorem of Schnorr [Sch78] that \( SAT \) is complete for NQL under \( \leq^l_\text{m} \). It is worth sketching Schnorr’s construction here (see also [BG93]) for reference below: Take a time-\( t(n) \) DTM \( M \) that decides a witness predicate \( R(x, y) \) for the given language \( A \in \text{NQL} \). Then as shown in [Sch76], \( M \) can be converted into a uniform family of \( O(t(n) \log t(n)) \)-sized circuits \( C_n \) of fan-in 2 in variables \( x_1, \ldots, x_n \) and \( y_1, \ldots, y_l \) such that for all \( x, x \in A \iff (\exists y_1, \ldots, y_l) C_n(x_1, \ldots, x_n, y_1, \ldots, y_l) = 1 \). Then assign a dummy variable to each of the \( O(n \log n) \) wires in \( C_n \) and write a 3-CNF formula that expresses that each output wire has the correct value given its input wires. This reduces \( A \) to \( SAT \) and is computable in time \( O(n \log n) \). The cases \( k > 1 \) follow by inserting this construction into the corresponding parts of the proofs for polynomial-time reductions in [Sto77, Wra77], similar to what we do in Proposition 2.4(a) below. Part (e) follows because the language \( QBF = \cup_1 B_k \) of quantified Boolean formulas belongs to quasilinear (in fact, linear) space.

Interestingly enough, we do not know whether \( QBF \) is complete for quasilinear space under quasilinear-time reductions. The standard reduction in [HU79], when applied to a given set \( A \) in \( \text{DSPACE}(O(n)) \), has a quadratic blowup in size. This seems related to the issue of whether Savitch’s simulation of nondeterministic space \( s(n) = \Omega(\log n) \) by deterministic space \( O(s(n)^2) \) must have quadratic blowup. By the same token, the familiar “one-line proof” that there is an oracle \( A \) making \( \text{NP}^A = \text{P}^A \), namely \( \text{NP}^{QBF} \subseteq \text{NPSPACE} = \text{PSPACE} = \text{P}^{QBF} \), is not valid for QL. However, the result (a) below is still true:

**Proposition 2.4** (a) \( \text{NQL}^{QBF} = \text{DQL}^{QBF} \).

(b) For any fixed quasilinear function \( q \), there is an oracle \( B \) such that \( \text{NQL}^B \) is not contained in \( \text{DTIME}[2^{\text{O}(n)}] \).

**Proof.** Let \( L \in \text{NQL}^{QBF} \), and let \( N \) be the oracle NTM that accepts \( L \). Let \( N' \) be an oracle NTM that on any input \( x \), guesses an accepting computation \( \vec{c} \) of \( N \). As before, the string \( \vec{c} \) includes the nondeterministic moves made by \( N \) and also lists \( y_1, \ldots, y_l \) of
queries answered positively and queries \( z_1, \ldots, z_m \) answered negatively. By Schnorr's construction, the condition that \( \overline{c} \) is an accepting computation can be encoded as a Boolean formula \( \phi_1 \) of quasilinear size. By the foregoing convention and lemmas on oracle queries, the condition that all the answers given in \( \overline{c} \) are correct can be represented by a Boolean formula \( \phi_2 \), which is just the conjunction of \( l + m \) instances of \( QBF \) and also has quasilinear size. Finally, a deterministic machine can in quasilinear time construct a formula \( \phi_x \) that is equivalent to \((\exists \overline{c})(\phi_1 \land \phi_2)\). Then \( x \in L \iff \phi_x \in QBF \). This shows in fact that \( L \) is in \( DQL_{QBF} \) with one query, and that \( QBF \) is complete for \( NQL_{QBF} \) under \( \leq_{sm} \).

Statement (b) holds by essentially the same standard construction which, given any polynomial \( p \), produces an oracle \( B \) such that \( NP^B \not\subseteq DTIME[2^{p(n)}] \).

The result of [PZ83] that \( \oplus \text{P}^P = \oplus \text{P} \) also carries over because of the quasilinear bound on the total length of all queries in an oracle computation: \( \oplus \text{QL} \subseteq \oplus \text{QL} \). However, it is unclear whether the theorem \( \text{BPP}^\oplus \text{P} = \text{BPP} [\text{Ko82}] \) carries over, because the amplification of success probability to \( 1 - 2^{-\text{polylog}} \) obtainable for \( \text{QPL} \) seems insufficient. However we are able to show, in the next section, that the well-known \( \text{NP} \subseteq \text{BP}[\oplus \text{P}] \) lemma from [VV86] and [Tod91] does carry over by a new construction, where all previous known constructions were quadratic or worse.

3 Quasilinear-Time Reduction to Parity

Let \( A \in \text{NP} \) with witness predicate \( R(x, y) \) and length bound \( q = q(n) \), and for any \( x \) let \( S_x := \{ y \mid \{0, 1 \}^q : R(x, y) \} \) be the corresponding witness set, so that \( x \in A \iff S_x \neq \emptyset \). Valiant and Vazirani [VV86] constructed a probabilistic NTM \( N \) that on any input \( x \) of length \( n \) first flips \( q^2 \)-many coins to form \( q \)-many vectors \( w_1, \ldots, w_q \) each of length \( q \). \( N \) also flips coins to form a number \( j, 0 \leq j \leq q \). Then \( N \) guesses \( y \in \{0, 1\}^q \) and accepts iff \( R(x, y) \) and for each \( i, 1 \leq i \leq j \), \( y \cdot w_i = 0 \), where \( \cdot \) is inner product of vectors over \( \text{GF}(2) \). Let \( N_{w,j} \) stand for the NTM \( N \) with \( w = w_1, \ldots, w_q \) and \( j \) fixed. Clearly whenever \( x \notin A \), for all \( w \) and \( i \), the number \( \# \text{acc}(N_{w,i}, x) \) of accepting computations of \( N_{w,i} \) on input \( x \) is zero. The basic lemma of [VV86] states that whenever \( x \in A \), \( \text{Pr}_w[(\exists j)(\# \text{acc}(N_{w,j}, x) = 1)] \geq 1/4 \). In particular, \( \text{Pr}_w[\# \text{acc}(N_{w,j}, x) \text{ is odd}] \geq 1/4(q + 1) \). A "product construction" yields an \( N' \) which flips coins to form just \( w \), guesses strings \( y_0, \ldots, y_q \), and achieves

\[
x \in A \implies \text{Pr}_w[\# \text{acc}(N'_w, x) \text{ is odd}] \geq 1/4,
\]

\[
x \notin A \implies \text{Pr}_w[\# \text{acc}(N'_w, x) \text{ is odd}] = 0
\]
for all $x$. In symbols, this says that $NP \subseteq RP[\oplus P]$ (cf. [Tod91]).

However, in the case $A = SAT$ addressed by [VV86], with $q(n) = n$, $N'$ runs in quadratic time—in fact, $N'$ flips quadratically many coins and makes quadratically many nondeterministic moves. It was well known that by using small families $\mathcal{H} = \{ H_k \}$ of universal hash functions [CW79] $h_k : \{0,1\}^q \rightarrow \{0,1\}^k$ ($1 \leq k \leq q + 1$) cuts the number $\nu(n)$ of random bits used to $2q(n)$. The construction of [CRS93] achieves the same effect, still with quadratic runtime when $q(n) = n$. Gupta [Gup93] gives a randomized reduction to parity which achieves constant success probability $3/16$ with only $\nu(n) = q(n)$ nondeterministic moves, but still using $q^2$-many random bits and quadratic time.

The only previous construction which ours does not improve by an order of magnitude in these measures is by Naor and Naor [NN90, NN93], which in this setting boils down to the following: Using $2q + 2$ coin flips, their $N$ determines, for each $k \leq q(n)$, a hash function $h_k \in H_k$. Next $N$ flips $q$ more coins to form $u \in \{0,1\}^{q+1}$. Then $N$ nondeterministically guesses $y \in \{0,1\}^q$ and $k$, $1 \leq k \leq q + 1$, and accepts iff $R(x,y) \land h_k(y) = 0 \land u_k = 1$. This uses $3q + 3$ random bits, achieves success probability at least $1/8$, and runs in the time to compute $h_k$, which is $O(q \log q \log \log q)$. Our construction achieves better constants, namely success probability arbitrarily close to $1/2$ and always using less than $2q$ random bits. Furthermore, it avoids the extra guess of $k$, and when applied to a given instance $\phi$ of SAT, yields a formula $\phi'$ of the simple form $\phi' = \phi \land \psi$.

Naor and Naor also mention error-correcting codes for similar purposes in passing, ascribing the idea to Bruck with a reference to [ABN+92]. However, using the codes in [ABN+92] appears to require computing exponentiation in finite fields $GF(2^m)$ where the size $m$ of field elements is polynomial in $n$. This is not known to be possible in quasilinear time, even by randomized algorithms, and the sequential method of von zur Gathen [vzG91] takes quadratic time on TMs. The main point of our construction is that by scaling down the size of the field, and using multi-variable polynomials, one can achieve quasilinear runtime. Our code is similar to those used in recent improvements of “holographic proof systems” [BLS91, Sed92], and is only inferior to that of [ABN+92] in using $2q - o(q)$ rather than $q + O(1)$ random bits.

### 3.1 Error-correcting codes

Let $\Gamma$ be an alphabet of size $2^l$. We can give $\Gamma$ the structure of the field $F = GF(2^l)$; then $\Gamma^n$ becomes an $n$-dimensional vector space over $F$. An $[N, K, D]$ code over $F$ is a set $C \subseteq \Gamma^n$ which forms a vector subspace of dimension $K$ (so $\|C\| = 2^K$), such that for all distinct $x, y \in C$, $d_H(x, y) \geq D$, where $d_H$ is Hamming distance. Since $C$ is closed under addition (i.e., a linear code), the minimum distance $D$ equals the minimum weight
(i.e., number of non-zero entries over $F$) of a non-zero codeword. The rate of the code is $R = K/N$, and the density is given by $\delta = D/N$. Any basis for $C$ forms a $K \times N$ generator matrix for the code. If $F = \text{GF}(2)$ we speak of a binary code. The following two examples form the main components of our construction:

- The Hadamard code $\mathcal{H}_k$ over $\{0,1\}$ of length $n = 2^k$ has $n$ codewords. The codewords can be arranged into an $n \times n$ array with rows and columns indexed by strings $u, v \in \{0,1\}^k$, and entries $u \cdot v$, where $\cdot$ is inner product over $\text{GF}(2)$. $\mathcal{H}_k$ has distance $d_k = 2^{k-1}$, so $\delta_k = 1/2$ is constant.

- The full $2^k$-ary generalized Reed-Muller code $\mathcal{R}_{2^d}(d,m)$ of order $d$, where $d < m(2^k-1)$, has length $N = 2^{km}$ over the field $F = \text{GF}(2^k)$. Each polynomial $f(x_1, \ldots, x_m)$, in $m$ variables over $F$ of total degree at most $d$, defines the codeword with entries $f(a_1, \ldots, a_m)$, where $\bar{a} = (a_1, \ldots, a_m)$ ranges over all sequences of arguments in $F$. In the important case $d \leq 2^k - 1$ a generator matrix for this code is easy to describe: it has one row for each monomial $x_1^{i_1}x_2^{i_2}\cdots x_m^{i_m}$ such that $i_1 + i_2 + \ldots + i_m \leq d$. Since $d \leq 2^k - 1$ these monomials are all distinct, and they are all linearly independent, so the dimension is $K = \binom{m+d}{d}$. The well-known property on which these codes are based (cf. [BFLS91, Sud92]) is that for every two distinct polynomials $f$ and $g$ over $F$ of total degree at most $d$, and for every $I \subseteq F$,

$$|\{ \bar{a} \in I^m : f(\bar{a}) = g(\bar{a}) \}| \leq d|I|^{m-1}. \quad (1)$$

With $I = F$, it follows that the density $\Delta$ is at least $1 - d/|F|$.

### 3.2 Application for reductions to parity

Let $\mathcal{R}(x,y)$ and $q(n)$ be a witness predicate and a quasilinear function that define the language $A$ as before. Suppose we have an allowance of $r(n)$ random bits, and desire success probability $\delta$. The idea is to find a $2^r \times 2^{r(n)}$ generator matrix $G$ for a binary code $C$ of constant density $\delta$. Then we can build a probabilistic NTM $N$ that works as follows:

1. Flip $r(n)$ coins to choose a column $j$.

2. Guess a row $i$, $1 \leq i \leq 2^q$, identified with a possible witness string $y_i \in \{0,1\}^q$.

3. Accept iff $\mathcal{R}(x,y_i) \land G(i,j) = 1$.

---

The standard notation is $\mathcal{R}_q(r,m)$ as in [TV91], where $q$ is a prime power and $r < m(q-1)$. Below $d = d_0 m$. 

---
Suppose $S = S_x$ is nonempty. Then to $S$ there corresponds the unique non-zero codeword $w_S := \sum_{y \in S} G(y, \cdot)$, where the sum is over $\text{GF}(2)$. Then $\# \text{acc}(N_j, x)$ is odd iff the $j$th entry of $w_S$ is a ‘1’. Since the proportion of non-0 entries of $w_S$ is at least $\delta$, $\Pr_j[\# \text{acc}(N_j, x) \text{ is odd}] > \delta$; that is, $N$ reduces $A$ to parity with success probability at least $\delta$. And if $S$ is empty, $N$ has no accepting computations at all.

Thus to show $\text{NQL} \subseteq \text{RQL}[\oplus \text{QL}]$, we need to construct a binary code $C$ so that

- Any selected entry $G(i, j)$ is computable in quasilinear time, and
- The density $\delta$ of $C$ is constant, the closer to $1/2$ the better.

In one level of coding over $\text{GF}(2)$, approaching $1/2$ from below is best possible, because by well-known results concerning the Plotkin bound in coding theory (see [MS77]), any binary code of density $1/2$ or more has too few elements to support the above application.

The generalized Reed-Muller code $R_{2^k}(d, m)$, which has length $N$ and density $\Delta$ over $\text{GF}(2^k)$, may instead be regarded as a binary code $R'$ of length $kN$ over $\text{GF}(2)$. But then we can only assert that the density of $R'$ is at least $\Delta/k$, because two distinct elements $a_1, a_2 \in \text{GF}(2^k)$ might differ in only one out of $k$ places as binary strings. The key idea, called concatenation of codes [For66], is to apply a second level of coding to these elements. In this case we take the so-called inner code to be the Hadamard code $H_k$. Then whenever $a_1 \neq a_2$ in $\text{GF}(2^k)$, $H_k(a_1)$ and $H_k(a_2)$ differ in at least half of their places as binary strings of length $2^k$. This results in a binary code $C$ of length $N2^k$ that has density $\Delta/2$. By arranging $\Delta > 1 - 2\epsilon$, as follows when $d/2^{k+1} < \epsilon$, one obtains the desired density $\delta > 1/2 - \epsilon$. The delicate part of the construction is to make $k$ large enough for the desired density, but not too large that the length $N2^k$ and time for operations in $\text{GF}(2^k)$ is prohibitive.

Let $\log^+ n$ abbreviate $\log n \log \log n \log \log \log n$.

**Theorem 3.1** For every language $A$ in $\text{NTIME}[q(n)]$, and any fixed $\epsilon < 1/2$, we can find a probabilistic parity machine $N$ that accepts $A$ with success probability $1/2 - \epsilon$, such that $N$ makes no more than $q = q(n)$ nondeterministic moves on inputs of length $n$, runs in time $O(n \log^+ n + q(n))$, and uses a number of random bits bounded by

$$2q - q \log \log q / \log q + (1 + \log(1/\epsilon))q / \log q + O(\log q).$$

**Proof.** On any input $x$, $N$ does the following:

1. $n := |x|, \; q := q(n)$
2. $b := \lceil \log_2 q \rceil \quad \text{/*block length for exponents*/}$
3. $d_0 := 2^b - 1 \quad \text{/*maximum degree in each variable*/}$
4. \( m := \lfloor q/b \rfloor \) /*number of variables*/

5. \( k := \lfloor \log_2 d_0 + \log_2 m + \log_2(1/\epsilon) - 1 \rfloor \)

6. Calculate an irreducible polynomial \( \alpha \) of degree \( k \) over \( GF(2) \)

7. Flip \( mk + k \) coins to form \( j = (a_1, \ldots, a_m, v) \), where \( v \in \{ 0, 1 \}^k \)

8. Guess \( y \in \{ 0, 1 \}^k \)

9. Taking \( b \) bits of \( y \) at a time, form integers \( i_1, i_2, \ldots, i_{m-1}, i_m \in \{ 0, \ldots, d_0 \} \). (It is OK for \( i_m \) to be truncated.)

10. Compute \( u := a_1^{i_1} \cdot a_2^{i_2} \cdots a_m^{i_m} \)

11. Compute \( G(y, j) := u \cdot v \) /*Hadamard code applied here*/

12. Accept iff \( R(x, y) \land G(y, j) = 1 \).

Steps 1–5 take linear time. It is now possible to do Step 6 deterministically in time that is polynomial in \( k \) (see [Sho88]), and since \( k \) is approximately \( \log q + \log n + \log(1/\epsilon) \), which is \( O(\log n) \) when \( \epsilon \) is fixed, the time for Step 6 is negligible. Step 7 takes time about \( nk/\log n \), which for fixed \( \epsilon \) is asymptotically less than the time \( q(n) \) for steps 8 and 9. For Step 10, we first note that to multiply two polynomials of degree \( k - 1 \) over \( GF(2) \) and reduce them modulo \( \alpha \) in the field \( GF(2^k) \) takes time \( t_1 = O(k \log k \log \log k) \) on standard Turing machine models (see [AHU74] and [Rab80]). The time to compute \( a^i \) in \( GF(2^k) \) where \( i \leq q \) is \( t_2 = O(\log q \cdot 2k \log k \log \log k) \) via repeated squaring, which is \( O(\log(n) \log^+ n) \). Thus the time for Step 10 is \( O(mt_2 + mt_1) = O(n \log^+ n) \). Step 11 takes negligible time, while step 12 takes another \( q(n) \) steps to compute \( R \). This yields the stated time bound. The random-bits bound follows on estimating \( mk + k \).

\[ \square \]

**Corollary 3.2** \( NQL \subseteq RQL[\oplus QL] \).

\[ \square \]

The first open problem is whether two or more alternations can be done in quasilinear time; that is, whether \( NQL^{NQL} \subseteq BQL[\oplus QL] \). The obstacle is the apparent need to amplify the success probabilities of the second level to \( 1 - 2^{-q} \), for which straightforward "amplification by repeated trials" takes time \( q^2 \). The second is whether the code can be improved and still give quasilinear runtime. Our codes have rate \( R = K/N = 2^{q/2^{2q-\ldots}} \), which tends to 0 as \( q \) increases. Families of codes are known for which \( R \) (as well as \( \delta \) stays bounded below by a constant; such (families of) codes are called good. Good codes require only \( q + O(1) \) random bits in the above construction. The codes in [ABN+92, JLJH92, She93] are good, but appear not to give quasilinear runtime here.
4 Search Versus Decision in Quasilinear Time

The classical method of computing partial, multivalued functions using sets as oracles is the \textit{prefix-set} method (cf. [Sel88]). To illustrate, let \( f \) be an arbitrary length-preserving, partial function from \( \Sigma^* \) to \( \Sigma^* \). Define:
\[
L_f = \{ x \# w \mid w \text{ is a prefix of some value of } f(x) \}.
\]

Clearly \( f \) is computable in quadratic time using \( L_f \) as an oracle. First we observe that for “random” functions \( f \), quadratic time is best possible.

\textbf{Theorem 4.1} There exist length-preserving functions \( f : \Sigma^* \rightarrow \Sigma^* \) with the property that there does not exist an oracle set \( B \) relative to which \( f \) is computable in less than \( n^2 - n \) steps.

\textbf{Proof.} Let \( B \) and an OTM \( M \) such that \( M^B(x) = f(x) \) on all strings \( x \in \{0,1\}^n \) be given, and suppose \( M^B \) runs in time \( g(n) \). Then the following is a description of \( f \) on \( \{0,1\}^n \):

- The finite control of \( M \), plus finite descriptions of the function \( g(n) \) and “this discussion” (see [LV90]). This has total length some constant \( C \).

- A look-up table for all the strings of length \( < n \) which belong to \( B \)—this is specifiable by a binary string of length \( \sum_{i=0}^{n-1} 2^i = 2^n - 1 < 2^n \).

- For each \( x \in \{0,1\}^n \), the answers given by \( B \) to those queries \( z \) made by \( M \) on input \( x \) such that \( |z| \geq n \). There are at most \( g(n)/n \) such queries. All of this is specifiable by a binary string of length \( 2^n g(n)/n \).

Now let \( K_f \) be the \textit{Kolmogorov complexity} of \( f \), relative to some fixed universal Turing machine. Then \( C + 2^n + 2^ng(n)/n \geq K_f \), so \( g(n) \geq nK_f/2^n - n - nC/2^n \). Since functions \( f : \{0,1\}^n \rightarrow \{0,1\}^n \) are in 1-1 correspondence with binary strings of length \( n2^n \), and (by simple counting) some such strings have Kolmogorov complexity at least \( n2^n \), there exist \( f \) with \( K_f \geq n2^n \). Then \( g(n) \geq n^2 - n \). \( \square \)

(Remarks: The \( n^2 - n \) is close to tight—an upper bound of \( g(n) \leq n^2 + 2n \log n \) is achievable by a modification of \( L_f \). By diagonalization one can also construct such functions \( f \) which are computable in exponential time.)

Hence the equivalence between functions and sets does not carry over to quasilinear time complexity in general. Theorem 4.1 can be read as saying that Kolmogorov-random functions have so much information that large query strings are needed to encode it.
We are interested in whether natural functions in NP, such as witness functions for NP problems, pack information as tightly.

Let $L$ be a language in NP and let $R$ be some polynomial-time witness predicate for $L$. Define the partial multivalued function $f_R$ by:

$$f_R(x) \mapsto y, \text{if } |y| = q(|x|) \text{ and } R(x,y).$$

Then $f_R$ is called a search function for $L$. The following is a straightforward extension of the standard notion of search reducing to decision in polynomial time [BD76, BBFG91, NOS93] to other time bounds $t(n)$.

**Definition 4.1.** Let $L \in \text{NP}$ and a time bound $t(n)$ be given. Then we say that search reduces to decision for $L$ in time $t(n)$ if there exists a witness predicate $R$ for $L$ and a $t(n)$ time-bounded deterministic oracle TM $M$ such that for all inputs $x$, if $x \in L$ then $M^L(x)$ outputs some $y$ such that $f_R(x) \mapsto y$, and if $x \notin L$ then $M^L(x) = 0$.

Let $\text{polylog } n$ abbreviate $(\log n)^{O(1)}$ as before. Then $\text{DTIME}[2^{\text{polylog } n}]$ is often referred to as quasi-polynomial time (cf. [Bar92]).

**Theorem 4.2** Let $L \in \text{NP}$. If search reduces to decision for $L$ in quasilinear time, then $L \in \text{DTIME}[2^{\text{polylog } n}]$.

**Proof.** Let $M$ be the oracle TM from Definition 4.1, and let $c, k$, and $n_0 \geq 4$ be constants such that for all strings $x$, $M$ on input $x$ halts within $cn \log^k |x|$ steps. For all inputs $x$ of length $< n_0$, whether $x \in L$ will be looked up in a table. For all $n \geq n_0$, define $f(n) = \lceil n / \log_3 n \rceil$.

Now we define a TM $M'$ that operates as follows on any input $x$ of length $n \geq n_0$: $M'$ simulates $M$ until $M$ makes some query $z$. If $|z| < n_0$, $M'$ answers from the table. If $|z| > f(n)$, we call $z$ a “large query.” Here $M'$ branches, simulating both a “yes” and a “no” answer to $z$. Finally, if $n_0 \leq |z| \leq f(n)$, then $M'$ calls itself recursively on input $z$ to answer the query. The above is a recursive description of what $M'$ does. The actual machine $M'$ simulates both the recursion and the branching on large queries using a stack, and halts and accepts iff at least one of the simulations of $M$ outputs a string $y$ such that $R(x,y)$ holds. Clearly $M'$ accepts $L$.

Let $t_{M'}(n)$ denote the running time of $M'$ on inputs of length $n$. We show that for all $n$, $t_{M'}(n) \leq 2^{c \log^{k+2} n}$. Since table-lookup takes only linear time, this holds for $n < n_0$. Now consider the binary tree $T$ whose nodes are large queries made by $M$, and whose edges represent computation paths by $M'$ between large queries. Then $T$ has depth at most $c \log^{k+1} n$ and at most $2^{c \log^{k+1} n}$ branches. The number of small queries on each branch is at most $cn \log^k n$, and each such query has length at most $n / \log n$. Hence
the time taken by $M'$ to traverse all branches, namely $t_{M'}(n)$, satisfies the following condition:

$$t_{M'}(n) \leq 2^{c \log^{k+1} n} \cdot cn \log^k n \cdot t_{M'}(n/\log n).$$

By induction hypothesis, $t_{M'}(n/\log n) \leq 2^{c(\log(n/\log n))^{k+2}}$. Substitution into (2) and some elementary calculation gives

$$t_{M'}(n) \leq 2^{c \log^{k+1} n} \cdot cn \log^k n \cdot 2^{c(\log n - \log \log n)^{k+2}} \leq 2^{c \log^{k+2} n}. \quad \Box$$

**Corollary 4.3** If search reduces to decision for SAT in quasilinear time, then

$$\text{NP} \subseteq \text{DTIME}[2^{\text{polylog} n}].$$

The technique of Theorem 4.2 extends (again with $f(n) = n/\log n$) to show that the quadratic bound on the search-to-decision reduction for SAT is likely to be optimal.

**Corollary 4.4** If there exists an $\epsilon > 0$ such that search reduces to decision for SAT in $\text{DTIME}[n^{1+\epsilon}]$, then for all $\delta > \epsilon$, $\text{NQL} \subseteq \text{DTIME}[2^{n^\delta}]$.

**Proof.** Following the last proof, again with $f(n) = n/\log n$, yields a timing analysis that places SAT into $\text{DTIME}[2^{n^{\log^2 n}}]$. The extra $\log^2 n$, and similar terms that arise with an arbitrary NQL language in place of SAT, can be absorbed into $2^{n^\delta}$ for any $\delta > \epsilon$. \quad \Box

Unlike the quasilinear case, we do not get the conclusion that all of NP is in $\text{DTIME}[2^{n^\delta}]$. If $A \in \text{NTIME}[n^k]$ then all we get from the above is a running time of $2^{n^{k\delta}}$, and the upper exponent is not bounded in $k$.

Stearns and Hunt [SH90] define a language $L \in \text{NP}$ to have power index $\epsilon$ if $\epsilon$ is the infimum of all $\delta$ such that $L \in \text{DTIME}[2^{n^\delta}]$. They classify familiar NP-complete problems according to known bounds on their power indices, and conjecture that SAT has power index 1. In this setting, Corollary 4.4 can be restated as:

**Corollary 4.5** If there exists an $\epsilon > 0$ such that search reduces to decision for SAT in $\text{DTIME}[n^{1+\epsilon}]$, then SAT has power index at most $\epsilon$.

This establishes a relation between reducing search to decision and the power index of an NP language. However, we now show that the converse is unlikely to be true.

Let EE stand for $\text{DTIME}[2^{2^{n^2(n)}}]$, and NEE for its nondeterministic counterpart. The classes EE and NEE were considered by Beigel, Bellare, Feigenbaum, and Goldwasser [BBFG91], and there are reasons for believing it unlikely that NEE = EE.

**Theorem 4.6** Suppose NEE $\neq$ EE. Then for all $k > 0$ there is a tally language in NP whose power index is at most $1/k$, but for which search does not reduce to decision in polynomial time.
Proof. Let $T$ be the tally set constructed in [BBFG91] such that search does not reduce to decision for $T$ in polynomial time, unless NEE = EE. Suppose $p$ is a polynomial such that for all $n$, all witnesses of the string $0^n$ are of length $p(n)$. Define:

$$T^k = \{0^{p(n)k} \mid 0^n \in T\}.$$ 

It is easy to see that $T^k$ has power index at most $1/k$, since an exhaustive search algorithm recognizes $T^k$ in time $2^{n^{1/k}}$. However if search reduces to decision in polynomial time for $T^k$, then it does so for $T$, which is a contradiction. 

\[
\Box
\]

5 Further Results and Connections to Other Work

The study of the relationship between search problems and decision problems is complicated by the fact that to a given language $A$ one can associate many different search problems, depending on the choice of witness predicate $R$ for $A$. The desire to find a property of decision problems alone that facilitates search led to several notions of helping proposed by Schöning [Sch85] and Ko [Ko87]. We extend their definitions from polynomial time to arbitrary time bounds $t(n)$ under our oracle convention. An oracle TM $M$ is robust if for every oracle $B$, $M$ with oracle $B$ halts for all inputs, and $L(M^B) = L(M^B)$. In other words, the language accepted by $M^B$ is the same for all oracles $B$.

Definition 5.1. A language $B$ 1-sided-helps a language $A$ in time $O(t(n))$ if there exist a robust oracle TM $M$ such that $L(M^{(1)}) = A$ and a constant $c \geq 1$ such that for all strings $x \in A$, $M^B(x)$ runs in time $ct(|x|)$.

The language $A$ is a self-1-helper in time $t(n)$ if $A$ 1-sided helps $A$ itself in time $t(n)$.

The point is that although the oracle $B$ doesn't affect the language accepted by $M$, it does enable strings in $A$ to be verified faster than might otherwise be the case. The robustness requirement rules out the oracle machine that simply queries its input $x$ to the oracle and similar trivialities. We write $O(t(n))$ rather than just $t(n)$ because linear speed-up does not hold for oracle machines. For polynomial time bounds, both Definition 4.1 and the notion of self-1-helping entail $A \in \text{NP}$, and we restrict attention to NP below. Balcázar [Bal90] proved that a language $A$ is a self-1-helper (in polynomial time) if and only if search reduces to decision for $A$ (in polynomial time). We observe first that Balcázar's proof carries over to any reasonable time bound $t(n)$.

Proposition 5.1 Let $A \in \text{NP}$. Search reduces to decision for $A$ in time $O(t(n))$ if and only if $A$ is a self-1-helper in time $O(t(n))$. 

14
Proof. ($\implies$): Suppose search reduces to decision for $A$ in time $t(n)$. By definition, there is a witness predicate $R$ for $A$ and a $t(n)$-time bounded deterministic oracle TM $M_0$ such that for all inputs $x$, if $x \in A$ then $M_0^A(x)$ outputs some $y$ such that $f_R(x) \rightarrow y$, and if $x \notin A$ then $M_0^A(x) = 0$. Define the robust TM $M$ as follows. On input $x$, $M$ simulates $M_0(x)$ for $t(|x|)$ steps. If a witness $y$ is produced, $M$ evaluates $R(x, y)$ and accepts $x$ if $R(x, y)$ holds; otherwise, $M$ performs a brute-force search for a witness $y$. Clearly $L(M^B) = A$ for all oracles $B$; moreover, for all $x \in A$, $M^A$ accepts $x$ in time at most $2t(n)$. It follows that $A$ is a self-1-helper in time $O(t(n))$.

($\Leftarrow$): Suppose $A$ is a self-1-helper in time $t(n)$; let $M$ be the robust TM such that $L(M, B) = A$ for all oracles $B$. Define the predicate $R(x, y) \equiv \text{"y is the sequence of oracle answers that causes $M$ to accept $x$ in time $t(n)$."}$ Clearly $R$ is a witness predicate for $A$ and $R \in \text{DTIME}[t(n)]$. Now let $M_0$ be a machine that on any input $x$ simulates $M^A(x)$ for $t(|x|)$ steps, and records the oracle responses as a string $y$. If $M^A$ halts and accepts within $t(|x|)$ steps, then $M_0$ outputs $y$; otherwise, $M_0$ outputs 0. Since the running time of $M_0(x)$ is bounded by $2t(n)$, we conclude that search reduces to decision for $A$ in time $2t(n)$. 

The above equivalence extends our results in the last section to the notion of self-1-helping. In particular, Theorem 4.2 can now be restated as:

**Theorem 5.2** Let $A \in \text{NP}$. If $A$ is a self-1-helper in quasilinear time, then $A \in \text{DTIME}[2^{\text{polylog} n}]$.

We can, however, prove a more general theorem in terms of 1-sided-helping, without the "self-" restriction. For some notation, let $P_{1,\text{help}}(C)$ and $\text{DQL}_{1,\text{help}}(C)$ denote, respectively, the classes of languages $A$ such that there is a language $B \in C$ which 1-sided-helps $A$ in polynomial time, respectively, in quasilinear time. Ko proved that for any nontrivial complexity class $C$, $\text{NP} = P_{1,\text{help}}(C)$ [Ko87]. In the quasilinear case, however, we observe that only one direction of Ko's result is likely to carry over.

**Theorem 5.3** (a) For all complexity classes $C$, $\text{DQL}_{1,\text{help}}(C) \subseteq \text{NQL}$.

(b) If $\text{NQL} \subseteq \text{DQL}_{1,\text{help}}(\text{NQL})$ then $\text{NP} \subseteq \text{DTIME}[2^{\text{polylog} n}]$.

**Proof.**

(a) Let $L \in \text{DQL}_{1,\text{help}}(A)$ for some language $A$. Let $M$ be the robust TM such that $L(M, B) = L$ for all oracles $B$, and such that $M^A$ accepts $L$ in time $q(n)$, where $q(n)$ denotes a quasilinear time bound. To show that $L \in \text{NQL}$, we build a nondeterministic TM $N$ that behaves as follows. On input $x$, $N$ simulates $M$ for exactly $q(|x|)$ steps;
whenever $M$ makes a query to the oracle, $N$ guesses the oracle response and continues its simulation. $N$ accepts $x$ if and only if $M$ accepts $x$ within $q(|x|)$ steps. If $x \in L$, then the path in which all oracle responses are guessed correctly is an accepting computation of $N$. If $x \not\in L$, it follows from the robustness of $M$ that no computation path of $N$ accepts $x$ (in any number of steps, much less in $q(|x|)$ steps). Thus we have $L(N) = L$.

(b) Suppose $SAT \in DQL_{1\text{-help}}(NQL)$. Then there exists some $A \in NQL$ that helps $SAT$ via a robust OTM $M$ in quasilinear time. Since $A \leq^t_m SAT$, $M$ can be replaced by a robust OTM $M'$ that makes $SAT$ 1-help itself in quasilinear time. The conclusion now follows via Theorem 5.2.

Next we consider the subject of bounded-query classes studied in [Bei87b, Bei87a, AG88, BGH89, ABG90, Bei91, BGGG93]. In particular, a language $L$ is defined to be P-supertese [Bei87a, ABG90] if for all $k$ and all oracles $B$, the function mapping a $k$-tuple of strings $x_1, \ldots, x_k$ to the $k$-tuple of answers $L(x_1), \ldots, L(x_k)$ cannot be computed in polynomial time while making at most $k - 1$ queries to $B$. Beigel, Kummer, and Stephan [BKS93] proved that $SAT$ is P-supertese iff $P \neq NP$. Their proof relativizes in this way: for any oracle $A$ such that $P^A \neq NP^A$, the language $SAT^A$ (or $K^A$ as below) is such that for all $k$ and languages $B$, every polynomial-time machine that solves $k$ instances of $L$ with oracle $0A \cup 1B$ must make at least $k$ queries to the $B$ half of the oracle. Since superteseness is another way of saying intuitively that the language $L$ packs information so tightly that no oracle can save on queries, one might suspect that it is closely related to our notion of search-to-decision requiring $n^2$-many query bits. However, this seems not to be so:

**Theorem 5.4** There exists an oracle $A$ such that $P^A \neq NP^A$, and search reduces to decision in quasilinear time for $SAT^A$.

This also gives a sense in which the quasi-polynomial upper bound for NP in Theorem 4.2 appears to be optimal. First we observe that a lemma of Selman [Sel79] carries over for quasilinear time reductions.

**Lemma 5.5** If $L_1$ and $L_2$ are such that $L_1 \equiv^t_m L_2$ and search reduces to decision in quasilinear time for $L_1$, then search reduces to decision in quasilinear time for $L_2$.

**Proof.** Let $g_1$ and $g_2$ be QL functions such that $L_1 \leq^t_m L_2$ via $g_1$ and $L_2 \leq^t_m L_1$ via $g_2$. Suppose $L_1$ helps itself via a robust OTM $M_1$ in quasilinear time $q(n)$. Then define $M_2$ to be a machine that on any input $x$ simulates $M_1$ on input $g_2(x)$, but when $M_1$ makes a query $z$, $M_2$ makes the query $g_1(z)$. Then $M_2$ is still a robust OTM, and $M_2$ with oracle $L_2$ on input $x$ has the same computation as $M_1$ with oracle $L_1$ on input $g_2(x)$. By
the lemmas in Section 2, \( M_2 \) with oracle \( L_2 \) still runs in quasilinear time. Thus \( L_2 \) is a self-1-helper in quasilinear time, and the conclusion follows via Proposition 5.1.

Proof. (of Theorem 5.4). Let \( \{Q_i\}_{i \in \mathbb{N}} \) be an enumeration of nondeterministic quasilinear time Turing machines. Then for any oracle \( X \), the following standard language is complete for NQL\(^X\) under \( \leq^{ui} \) reductions:

\[
K^X = \{ (x, i, y) : Q_i \text{ accepts } x \text{ within } |y| \text{ steps } \}.
\]

For convenience, we modify \( K^X \) into another NQL-complete language \( \text{pad}K^X \) such that the lengths of all strings in \( \text{pad}K^X \) are powers of 2:

\[
\text{pad}K^X = \{ (x, i, y, 0^r) : r < |(x, i, y)|, |(x, i, y, 0^r)| \text{ is a power of 2, and } (x, i, y) \in K^X \}.
\]

It is clear that \( K^X \) reduces to \( \text{pad}K^X \) in linear time, so that \( \text{pad}K^X \) is NQL\(^X\)-complete. Now define a function \( e \) such that for all integers \( n \), if \( n + \lceil \log^2 n \rceil \) is even, then \( e(n) = n + \lceil \log^2 n \rceil \), else \( e(n) = n - 1 + \lceil \log^2 n \rceil \). In other words, for all \( n \), \( e(n) \) is the largest even integer smaller than or equal to \( n + \lceil \log^2 n \rceil \).

The following language is in NP\(^X\) for all oracles \( X \):

\[
L^X = \{ 0^n : n \text{ is odd and } \Sigma^n \cap X \neq \emptyset \}.
\]

We will construct an oracle \( A \) such that the following conditions are satisfied.

(i) \( L^A \in \text{NP}^A - \text{P}^A \), and

(ii) for all \( u \in \Sigma^* \), \( u \in \text{pad}K^A \) iff \( u \) is a prefix of a string \( v \in A \) such that \( |v| = e(|u|) \).

The oracle \( A \) is constructed in stages. At stage \( n \), we decide the membership of all strings of length \( n \). Also at stage \( n \), some strings of length \( > n \) may be reserved for \( \overline{A} \), and some strings may be added to \( A \). If nothing is done at stage \( n \), then all strings of length \( n \) belong to \( \overline{A} \). No decisions on membership are ever changed, and no requirements are "injured."

Now let \( \{M_i\}_{i \in \mathbb{N}} \) be an enumeration of deterministic polynomial-time oracle TMs, each \( M_i \) with polynomial running time \( p_i \). An index \( i \) will be canceled if and when we ensure that \( M_i^A \) does not recognize \( L^A \). As usual, \( A(n) \) denotes the strings in \( A \) prior to stage \( n \). Let \( A(0) = \emptyset \). In keeping with the definition of \( \text{pad}K^A \), we focus on lengths \( \ell \) that are a power of 2.

Stage \( n = 2m \): If there does not exist a power-of-2 \( \ell \) such that \( n = e(\ell) \), then do nothing and go to the next stage. Else, for every string \( z \in \Sigma^n \) that has not been reserved for \( \overline{A} \) at an earlier stage, determine the unique string \( x \) of length \( \ell \) that is a prefix of \( z \). Add \( z \) to \( A \) if and only if \( x \) is in \( \text{pad}K^{A(n)} \).
Stage $n = 2m + 1$: If there exists a string of length $\geq n$ that has been reserved for $\overline{A}$ at some previous stage, do nothing and go to stage $n + 1$. Else, let $i$ be the least uncanceled index. If there exists a power-of-2 $\ell$ such that $n + 1 = e(\ell)$ and $p_i(n) < 2^{\log^2 \ell}$, then $i$ can be canceled at this stage; if not, do nothing and go to stage $n + 1$. In the course of canceling $i$, run $M_i^{A(n)}(0^n)$, and reserve for $\overline{A}$ all strings of length greater than or equal to $n$ that are queried during this computation. Then if $M_i^{A(n)}(0^n)$ rejects, add some unreserved string of length $n$ to $A$. Finally cancel $i$ and go to stage $n + 1$. This ends the construction. We now prove that $A$ satisfies conditions (i) and (ii).

Claim 5.6 $L^A \in NP^A - P^A$.

Proof. It suffices to prove that every index $i$ is eventually canceled. Suppose not; then there is a least uncanceled index $i$. Let $n_0$ be a stage by which all indices less than $i$ have been canceled. At all odd stages $n > n_0$, index $i$ is the only one that can be acted upon, so it suffices to show that the conditions for canceling $i$ hold infinitely often. For all sufficiently large $n$, $p_i(n) < 2^{\log^2 n}$, and for infinitely many odd $n$, there exists a power-of-2 $\ell_n$ such that $n + 1 = e(\ell_n)$. Hence for infinitely many odd $n$,

$$p_i(n) < 2^{(\log^2 (\ell_n + \log^2 \ell)} < 2^{\log^2 (\ell + \log^2 (\log^2 \ell)} < 2^{2\log^2 \ell}.$$

Since no action at even stages reserves any string for $\overline{A}$, there must be some odd stage after $n_0$ when $i$ is canceled.

Next, we demonstrate that the oracle construction satisfies condition (ii).

Claim 5.7 For all strings $x$,

$$x \in \text{pad}K^A \iff (\exists z)[z \in \Sigma^{e(|x|)} \cap A \text{ and } x \text{ is a prefix of } z].$$

Proof. It suffices to show that for all powers-of-2 $\ell$ and all strings $x \in \Sigma^\ell$, there exists a string $v \in \Sigma^{\log^2 \ell}$ such that $xv$ is never reserved for $\overline{A}$. Then the construction at stage $e(\ell)$ ensures the claim.

We first note that the restriction of $e$ to powers of 2 is a one-to-one function, as follows because $\log^2 \ell$ is an integer. By construction, strings of length $e(\ell)$ can be added to $A$ only at stage $e(\ell)$. Suppose $i$ is the last index that was canceled before stage $e(\ell)$ of the construction, and let $n_i$ be the stage at which $i$ was canceled. Then there is a unique power-of-2 $\ell_i$ such that $e(\ell_i) = n_i + 1$, and also $\ell_i < \ell$. Since $n_i$ was the last canceling stage, and $n_i < e(\ell)$, any string of length $e(\ell)$ that was reserved for $\overline{A}$ before step $e(\ell)$ was reserved at stage $n_i$. From the fact that $i$ was canceled, $p_i(n_i) < 2^{\log^2 n_i}$. Hence the number of strings of length $e(\ell)$ that were reserved for $\overline{A}$ at stage $n_i$ is at most $p_i(n_i)$, because no other indices were canceled between stages $n_i$ and $e(\ell)$. Then
\( p_i(n) < 2^{\log^2 \ell_i} < 2^{\log^2 \ell} \). Since there are \( 2^{\log^2 \ell} \)-many strings \( u \) such that \( xu \in \Sigma^e \), at least one such string is never reserved for \( \overline{A} \). This proves the claim. \( \square \)

It remains to show that search reduces to decision for \( K^A \) in quasilinear time. We first consider the following NQL language

\[
\text{prefix } K^A = \{ x \# u : (\exists u \in A) \left[ |u| = e(|x|) \right. \text{ and } x \# u \text{ is a prefix of } u \}. \]

By the construction of \( A \), it follows that \( \text{pad } K^A \) reduces in linear time to \( \text{prefix } K^A \), and since \( \text{pad } K^A \) is NQL-complete, \( \text{prefix } K^A \) is also NQL-complete. Since witnesses to membership of a string \( x \) in \( \text{prefix } K^A \) are of length \( |x| + \log^2 |x| \), the total number of steps performed by the standard prefix search algorithm is bounded by

\[
O(|n| + (|n| + 1) + (|n| + 2) + \cdots + (|n| + \log^2 |n|)) = O(n \log^2 n).
\]

The theorem now follows by Lemma 5.5, since \( \text{prefix } K^A \equiv_{t_1} \text{pad } K^A \equiv_{t_1} \text{SAT}^A \). \( \square \)

It suffices to take \( e(n) \) to be any function that has a higher order of growth than \( n \log n \). However, this leaves open the question of whether \( P = \text{NP} \) can be shown to follow if \( \text{SAT} \) helps itself in time \( O(n \log n) \). This aside, Theorem 5.4 really does pertain to quasilinear time, not just to linear time or time \( O(n \log n) \).

6 Conclusions and Further Research

One large source of interest is that we have identified a new hypothesis to the effect that \( \text{NP} \)-complete sets, and \( \text{SAT} \) in particular, not only lie outside \( P \), but also pack their hardness very tightly. Our hypothesis is the last on the following list:

(a) \( \text{SAT} \) has power index 1 [SH86].

(b) \( \text{SAT} \) is \( P \)-superterse [Bei87a].

(c) The search function for \( \text{SAT} \) does not belong to \( P^{\text{NP}[o(n)]} \) [Kre88].

(d) \( \text{NP} \) does not have \( p \)-measure zero in exponential time [Lut93]

(e) The search function for \( \text{SAT} \) requires \( \Omega(n^2) \) query bits to compute in polynomial time, with any oracle set (or at least any oracle set in NQL).
It would be interesting to seek closer relationships among these hypotheses. We have
given some oracle evidence that (e) is a stronger assertion than (b), and we have shown
that (a) implies (e). Krentel showed that the search functions for SAT and the NP-
complete MaxClique problem do not belong to PF^{NP[O(log n)]} unless P = NP. There
has been considerable interest in whether these functions can be shown to be outside
PF^{NP[O(n^{2})]} or even PF^{NP[O(log^2 n)]} unless P = NP. Theorem 4.2 provides a viewpoint
on this question: if the largest clique can be found in PF^{NP} using at most n polylog n query
bits, then NP ⊆ DTIME[2^{polylog n}], and n^{1+c} query bits would place MaxClique into
DTIME[2^{n^c}]. The closest impact of (e) may be in relation to (d). By results of Juedes
and Lutz [JL93], (d) implies that there exists ϵ > 0 such that SAT does not have power
index ϵ, hence that search does not reduce to decision for SAT in time O(n^{1+c}). We
believe there should be deeper connections.

Another important question concerns the existence of “QL one-way” functions. Do
there exist length-preserving 1-1 functions f which are computable in qlin time but not
invertible in qlin time? Homer and Wang [HW89] construct, for any k ≥ 1, functions
computable in quadratic time which are not invertible in time O(n^{k}), but their methods
seem not to apply for qlin time or length-preserving functions. If DQL ≠ UQL, then QL
one-way functions exist, but unlike the polynomial case (assuming P ≠ UP), the converse
is not known to hold. It may even be possible to construct an oracle A such that QL^{A}
one-way functions exist, and yet DQL^{A} = NQL^{A}. We look toward further research which
might show that length-preserving functions with certain “pseudorandom” properties
cannot be inverted in qlin time, unless unlikely collapses of complexity classes occur.

Acknowledgments We thank Madhu Sudan for interactive assistance during FCRC’93
in tuning the parameters of the code in Section 3. We thank Stephen Bloch and Michael
Loui for comments on earlier versions of this paper, and Jonathan Buss, Judy Goldsmith,
Jack Lutz, and Kripa Sundar for pertinent discussions.

References

[ABG90] A. Amir, R. Beigel, and W. Gasarch. Some connections between bounded query
classes and nonuniform complexity. In Proc. 5th Annual IEEE Conference on Struc-


