On Quasilinear Time Complexity Theory

Ashish V. Naik[1]
SUNY Buffalo
avnaik@cs.buffalo.edu

Kenneth W. Regan[2]
SUNY Buffalo
regan@cs.buffalo.edu

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

[1]Supported in part by NSF grant CCR-9002292.
[2]Supported in part by NSF grant CCR-9011248.

Abstract

This paper furthers the study of quasi-linear time complexity initiated by Schnorr and Gurevich and Shelah. 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 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 ($\leq_m^{ql}$). 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 $\leq_m^{ql}$, so that the NQL vs. DQL question takes on much the same shape as NP vs. P. (Throughout this paper, $\log n$ stands for the real-valued logarithm to base 2. When it is important to make the value an integer, we write $\lfloor \log n \rfloor$ or $\lceil \log n \rceil$ accordingly.)

One theoretical difficulty with the concept of quasilinear 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 deterministic quasilinear time. They also showed that nondeterministic *qlin* time for these machines equals NQL; i.e., the nondeterministic RAM variants are no more powerful than nondeterministic Turing machines for *qlin* time. However, currently it appears that the deterministic machines in [GS89] accept more languages than those in the deterministic Turing machine class DQL. Moreover, for all $d > 1$, Turing machines with d-dimensional tapes may accept more languages in time *qlin* than do TMs with (d–1)-dimensional tapes. Graedel [Gra90] studied the class of languages L such that for all $\epsilon > 0$, L is acceptable in time $O(n^{1+\epsilon})$ by one of the respective kinds of machines, observing a slightly better robustness picture. (For background on these machines and simulations, see [WW86, vEB90].) Our answer to this problem of non-robustness is to arrange that all of our quasilinear-time upper bounds be attainable by Turing machines, and that our lower bounds hold even for RAMs.

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 quasilinear time? Section 2 observes that the basic results for the polynomial hierarchy hold also for the quasilinear 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 quasilinear 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 quasilinear 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 quasilinear 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$[2^{\mathrm{polylog}\, n}]$. 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" [Sch85b, 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 $\mathrm{NP}^A \neq \mathrm{P}^A$, so that *SAT* relative to A is still "P^A-superterse" (see [BKS93]). This also gives evidence that our quasipolynomial simulation of NP in Theorem 4.2 is close to optimal. The oracle construction is related to ones by Kintala and Fischer [KF80] on classes defined by polynomial-time NTMs allowed to make at most $O(\log^k n)$ nondeterministic moves; for other related work on "limited nondeterminism," see [BG93, BG94]. 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 $y_1, \dots y_m \in \Sigma^*$ such that $\sum_{i=1}^m |y_i| = r$, let $y = \langle y_1, \dots, y_m \rangle$ stand for the binary string of length $2r + 2m$ 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 $(\exists^q y)$ for $(\exists y \in \{0,1\}^{q(n)})$, $(\forall^q y)$ for $(\forall y \in \{0,1\}^{q(n)})$. The notation $(\#^q y : R(x,y))$ means "the number of strings $y \in \{0,1\}^{q(|x|)}$ such that $R(x,y)$ holds." The following generalizes a standard notion to other time bounds.

Definition 2.1. A *witness predicate* for a language A is any binary predicate R such that $A = \{\, x : (\exists y)\, R(x,y) \,\}$. If $R \in \mathrm{P}$ and there is a polynomial p such that $A = \{\, x : (\exists^p y)\, R(x,y) \,\}$, then we call R a *polynomial witness predicate*; while if $R \in \mathrm{DQL}$ and there is a quasilinear function q such that $A = \{\, x : (\exists^q y)\, R(x,y) \,\}$, then R is a *quasilinear witness predicate*.

Now we note the following provision about oracle Turing machines M made standard in both [WW86] and [BDG88] (see also [LL76, Wra76, 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 A and B are languages such that $L(M^B) = A$ and M^B runs in quasilinear time, then we write $A \leq_T^{ql} B$. As usual we may also write $A \in \mathrm{DQL}^B$ or $A \in \mathrm{DQL}(B)$, and if M is nondeterministic, $A \in \mathrm{NQL}^B$ or $A \in \mathrm{NQL}(B)$. 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, letting q stand for a quasilinear function:

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

(b) $A \in \mathrm{UQL}[R]$ if there exists q such that for all $x \in \Sigma^*$,

$$\begin{aligned} x \in A &\implies (\#^q y : R(x,y)) = 1, \quad \text{and} \\ x \notin A &\implies (\#^q y : R(x,y)) = 0. \end{aligned}$$

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

(d) $A \in \mathrm{BQL}[R]$ if there exists q such that for all $x \in \Sigma^*$,

$$\begin{aligned} x \in A &\implies (\#^q y : R(x,y))/2^q > 2/3, \quad \text{and} \\ x \notin A &\implies (\#^q y : R(x,y))/2^q < 1/3. \end{aligned}$$

(e) $A \in \mathrm{RQL}[R]$ if there exist q and $\epsilon > 0$ such that for all $x \in \Sigma^*$,

$$\begin{aligned} x \in A &\implies (\#^q y : R(x,y))/2^q > 2/3, \quad \text{and} \\ x \notin A &\implies (\#^q y : R(x,y)) = 0. \end{aligned}$$

For any class $\mathcal{C}$ of languages, $\mathrm{NQL}[\mathcal{C}]$ equals $\cup_{R \in \mathcal{C}} \mathrm{NQL}[R]$, and similarly for the other operators. With $\mathcal{C} = \mathrm{DQL}$ these classes are simply written NQL, UQL, $\oplus$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 BQL[$\mathcal{C}$] and RQL[$\mathcal{C}$] provided $\mathcal{C}$ is closed downward under "polylogarithmic majority truth table reductions." (A language A polylog-majority truth-table reduces to a language B if there exist $k > 0$ and a polynomial-time computable function f such that for all but finitely many x, $f(x)$ is a set S of at most $(\log |x|)^k$-many strings that satisfies $x \in A \iff$ more than half of the members of S belong to B.) This is also enough to give BQL[BQL[$\mathcal{C}$]] = BQL[$\mathcal{C}$].

Definition 2.3. The *quasilinear time hierarchy* is defined by: $\Sigma_0^{ql} = \Pi_0^{ql} = \Delta_0^{ql} = \mathrm{DQL}$, and for $k \geq 1$,

$$\Sigma_k^{ql} = \mathrm{NQL}[\Pi_{k-1}^{ql}], \quad \Pi_k^{ql} = \text{co-}\Sigma_k^{ql}, \quad \Delta_k^{ql} = \mathrm{DQL}^{\Sigma_{k-1}^{ql}}.$$

Also QLH := $\cup_{k=0}^{\infty} \Sigma_k^{ql}$, and QLSPACE := DSPACE[$qlin$]. By the results of [GS89], all these classes from NQL upward are the same for Turing machines and log-cost RAMs.

The proofs of *Cook's Theorem* in several standard references [Coo71, GJ79, HU79, Pap94] Schnorr [Sch78] showed that the formula size and running time of *Cook's Theorem* applied to a time-$q(n)$ nondeterministic TM N, which is quadratic or cubic in $q(n)$ in the proofs given by several standard references [Coo71, GJ79, HU79, Pap94], can be brought down to $O(q(n) \log q(n))$. For later reference we give a brief sketch of Schnorr's construction, following Buss and Goldsmith [BG93]. Given N, form the time-$q(n)$ deterministic TM M such that for all x, $x \in L(N) \iff (\exists^q y)[M$ accepts $\langle x, y\rangle]$. Then as shown in an earlier paper by Schnorr [Sch76], M can be converted into a uniform family of $O(q(n) \log q(n))$-sized circuits C_n of fan-in 2 in variables $x_1, \ldots, x_n$ and $y_1, \ldots, y_q$ such that for all x, $x \in L(N) \iff (\exists y_1, \ldots, y_q)\, C_n(x_1, \ldots, x_n, y_1, \ldots, y_q) = 1$. Now assign a dummy variable to each of the $O(q(n) \log q(n))$ 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 $L(N)$ to SAT and is computable in time $O(q(n) \log q(n))$.

which is again quasilinear.

Next we observe the following concavity property of quasilinear functions.

Lemma 2.1 *(a) Let $q(n) = cn \log^k n$, let $n_1, \ldots, n_m \geq 1$, and let $r = \sum_{i=1}^m n_i$. Then $\sum_{i=1}^m q(n_i) \leq q(r)$.*

(b) If $q(n) = cn \log^k n + d$, 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 $m - 1$, $\sum_{i=1}^m q(n_i) \leq q(r - n_m) + q(n_m)$. Define the real function $Q(x) = q(r - x) + q(x)$. For $k \geq 2$, the second derivative of Q with respect to x equals

$$ck(\log e)\left[\frac{(\log e)(k–1)\log^{k-2}x}{x}+\frac{\log^{k-1}x}{x}+\frac{\log^{k-1}(r-x)}{r-x}+\frac{(\log e)(k–1)\log^{k-2}(r-x)}{r-x}\right].$$

This is positive for all real x such that $1 < x < r–1$, so the maximum value of $Q(x)$ on the closed interval $[1, r–1]$ is attained at one of the endpoints. This value equals $q(r–1)$, which is less than $q(r)$. Since $n_m \geq 1$ and $r - n_m \geq 1$, the conclusion follows. (Indeed, for $m \geq 2$ we have $\sum_{i=1}^{m} q(n_i) < q(r)$.)

(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''. $\square$

Corollary 2.2 *The relation $\leq_T^{ql}$ is transitive. In particular,* $\mathrm{DQL}^{\mathrm{DQL}} = \mathrm{DQL}$.

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. $\square$

With this in hand it is straightforward to show that the most fundamental properties of the polynomial hierarchy (from [Sto76, Wra78]) carry over to QLH.

Theorem 2.3 *(a) (Equivalence of oracles and quantifiers): For all $k \geq 1$, $\Sigma_k^{ql} = \mathrm{NQL}^{\Sigma_{k-1}^{ql}}$.*

(b) (Upward collapse): For all $k \geq 0$, if $\Sigma_k^{ql} = \Pi_k^{ql}$ then $\mathrm{QLH} = \Sigma_k^{ql}$.

(c) (Turing closure): For all $k \geq 0$, $\Sigma_k^{ql} \cap \Pi_k^{ql}$ is closed downward under $\leq_T^{ql}$.

(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 Σ_k^{ql} under DQL *many-one reductions.*

(e) $\mathrm{QLH} \subseteq \mathrm{QLSPACE}$.

Proof. (a) The base case $k = 1$ follows via $\mathrm{NQL}^{\mathrm{DQL}} = \mathrm{NQL}[\mathrm{DQL}^{\mathrm{DQL}}] = \mathrm{NQL}[\mathrm{DQL}] = \mathrm{NQL}$. The induction case for $k > 1$ is typified by showing that $\mathrm{NQL}^{\mathrm{NQL}} \subseteq \Sigma_2^{ql}$. Let the oracle NTM N accept L with oracle $A \in \mathrm{NQL}$ in quasilinear time $r(n)$. Without loss of generality, we may suppose that N does not write a query bit and make a nondeterministic move in the same step. There is a DQL predicate R and a quasilinear function q such that for all $y \in \Sigma^*$, $y \in A \iff (\exists^q v)\ R(y, v)$. Let $q'(n) = q(r(n))$ for all n. Then for all $x \in \Sigma^*$,

$$x \in L \iff (\exists^r \vec{c})(\exists^{q'} \vec{v})(\forall^{q'} \vec{w})\ Matrix(x, \vec{c}, \vec{v}, \vec{w}),$$

where $Matrix(x, \vec{c}, \vec{v}, \vec{w})$ states the following: $\vec{c}$ 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(b). Since $Matrix(x, \vec{c}, \vec{v}, \vec{w})$ is decidable in quasilinear time, this is a Σ_2^{ql} 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 $\leq_m^{ql}$. It is worth sketching Schnorr's construction here (see also [BG93]) for reference below: Given $A \in$ NQL, there are quasilinear functions q, r and a DTM M such that for all x, $x \in A \iff (\exists^q y)[M \text{ accepts } \langle x, y\rangle]$, where for all x and y, $M(\langle x, y\rangle)$ halts within $r(|x|)$ steps. Then as shown in [Sch76], for all n, M can be converted into a uniform family of $O(r(n) \log r(n))$-sized circuits C_n of fan-in 2 in variables $x_1, \ldots, x_n$ and $y_1, \ldots, y_q$ such that for all x, $x \in A \iff (\exists y_1, \ldots, y_q)\, C_n(x_1, \ldots, x_n, y_1, \ldots, y_q) = 1$. Then assign a dummy variable to each of the $O(r(n) \log r(n))$ 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 $O(r(n) \log r(n))$, which is again quasilinear. The cases $k > 1$ follow by inserting this construction into the corresponding parts of the proofs for polynomial-time reductions in [Sto76, Wra78], similar to what we do in Proposition 2.4(a) below. Part (e) follows because the language $QBF = \cup_k B_k$ of quantified Boolean formulas belongs to quasilinear (in fact, linear) space. $\square$

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 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 $\mathrm{NP}^A = \mathrm{P}^A$, namely $\mathrm{NP}^{QBF} \subseteq \mathrm{NPSPACE} = \mathrm{PSPACE} = \mathrm{P}^{QBF}$, is not valid for QL. However, the result (a) below is still true:

Proposition 2.4 *(a) There exists an oracle A such that* $\mathrm{NQL}^A = \mathrm{DQL}^A$.

(b) There exists an oracle B such that not only is $\mathrm{NQL}^B \neq \mathrm{DQL}^B$*, but also for any fixed quasilinear function q,* NQL^B *is not contained in* $\mathrm{DTIME}^B[2^{q(n)}]$.

Proof. (a) Tretkoff [Tre86] showed that if one takes $A := \{\langle M, x, 0^n\rangle :$ the DTM M accepts x in space$n\}$, then $\mathrm{DLIN}^A = \mathrm{NLIN}^A = \mathrm{LINSPACE}^A$ (using the above oracle convention). By the same token we note that $\mathrm{DQL}^A = \mathrm{NQL}^A$: Let $L \in \mathrm{NQL}^A$ via an OTM N that runs in quasilinear time $q(n)$. Let M' be a non-oracle deterministic TM with some tapes devoted to simulating all branches of N, and others devoted to answering all oracle calls made by N. Since N runs in $q(n)$ *time*, it cannot write any queries of length $> q(n)$. Since the space overhead for simulating the NTM N and for universal simulation of DTMs M is linear, M' runs in $q'(n) = O(q(n))$ space. Hence a DQL-machine M'' on input x can write down the query $\langle M', x, 0^{q'(|x|)}\rangle$ to A, and in fact, $L \leq_m^{ql} A$.

(b) This follows by inter-twining over all $q(n)$ the standard construction of an oracle B such that $L_q := \{\, 0^n : B \cap \Sigma^{q(n)} \neq \emptyset \,\}$ is not in $\mathrm{DTIME}^B[q(n)]$. □

Remark: In (a) one can also take $A = QBF$: Let $L \in \mathrm{NQL}^{QBF}$ via the oracle NTM N. Let N' be an oracle NTM that on any input x, guesses an accepting computation $\vec{c}$ of N. 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 $\vec{c}$ is an accepting computation can be encoded as a Boolean formula ϕ_1 of quasilinear size. By the foregoing convention and lemmas on oracle queries, the condition that all the answers given in $\vec{c}$ are correct can be represented by a Boolean formula ϕ_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 ϕ_x that is equivalent to $(\exists \vec{c})(\phi_1 \wedge \phi_2)$. 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 $\leq_m^{ql}$. The important difference from the polynomial case is that this appears not to work if N is quasilinear *space* bounded, even if N is deterministic.)

The result of [PZ83] that $\oplus\mathrm{P}^{\oplus\mathrm{P}} = \oplus\mathrm{P}$ also carries over because of the quasilinear bound on the total length of all queries in an oracle computation: $\oplus\mathrm{QL}^{\oplus\mathrm{QL}} = \oplus\mathrm{QL}$. However, it is unclear whether the theorem $\mathrm{BPP}^{\mathrm{BPP}} = \mathrm{BPP}$ [Ko82] carries over, because the amplification of success probability to $1 - 2^{-\,\mathrm{polylog}}$ obtainable for BQL seems insufficient. However, we show in the next section that the well-known $\mathrm{NP} \subseteq \mathrm{BP}[\oplus\mathrm{P}]$ 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 $A \in \mathrm{NP}$ with witness predicate $R(x,y)$ and length bound $q = q(n)$, and for any x let $S_x := \{\, y \in \{\,0,1\,\}^q : R(x,y)\,\}$ 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 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 $\#acc(N_{w,j},x)$ of accepting computations of $N_{w,j}$ on input x is zero. The basic lemma of [VV86] states that whenever $x \in A$, $\Pr_w[(\exists j)\#acc(N_{w,j},x) = 1] \geq 1/4$. In particular, $\Pr_{w,j}[\#acc(N_{w,j},x) \text{ is odd}] \geq 1/4(q+1)$. A "product construction" yields an N' which flips coins to form just w, guesses strings $y_0,\ldots,y_q$, and achieves

$$\begin{aligned} x \in A &\implies \Pr_w[\#acc(N'_w,x) \text{ is odd}] \geq 1/4, \\ x \notin A &\implies \Pr_w[\#acc(N'_w,x) \text{ is odd}] = 0 \end{aligned}$$

for all x. In symbols, this says that $\mathrm{NP} \subseteq \mathrm{RP}[\oplus\mathrm{P}]$ (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 observed in [BCGL89] that one can use small families $\mathcal{H} = \{\, H_k \,\}$ of *universal*$_2$ ([CW79]) hash functions for the Valiant-Vazirani reduction, and using such a family $h_k : \{\,0,1\,\}^q \to \{\,0,1\,\}^k$ $(1 \leq k \leq q+1)$ cuts the number $r(n)$ 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.

Subsequent to our finding the application of error-correcting codes to make the time quasilinear and $r(n) < 2n$, we discovered that a trick of Naor and Naor [NN90, NN93] can also be applied to this reduction: Build a probabilistic NTM N that first uses $2q+2$ coin flips to determine, for each $k \leq q(n)+1$, a hash function $h_k \in H_k$. Next N flips $q+1$ 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 $R(x,y) \;\wedge\; h_k(y) = 0 \;\wedge\; u_k = 1$. This uses $3q+3$ random bits, achieves success probability at least 1/8, and runs in the time to compute h_k, which is $O(q \log q \operatorname{loglog} q)$. 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 ϕ of SAT, yields a formula ϕ' of the simple form $\phi' = \phi \;\wedge\; \psi$.

Naor and Naor also mention error-correcting codes for similar purposes in-passing, ascribing the idea to Bruck with a reference to [ABN$^+$92]. However, using the codes in [ABN$^+$92] appears to require computing exponentiation in finite fields GF(2^m) where the size m of field elements is polynomial in n. This is not known to be possible in quasilinear time, even by randomized algorithms, and the sequential method of von zur Gathen [vzG91] takes quadratic time on TMs. The main point of our construction is that by scaling down the size of the field, and using multi-variable polynomials, one can achieve quasilinear runtime. Our code is similar to those used in recent improvements of "holographic proof systems" [BFLS91, Sud92], and is only inferior to the code of [ABN$^+$92] in using nearly $2q$ rather than $q + O(1)$ random bits.

3.1 Error-correcting codes

Let Γ be an alphabet of size 2^l. We can give Γ the structure of the field $F = \mathrm{GF}(2^l)$; then Γ^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 $\|C\| = 2^{lK}$), 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 $F = \mathrm{GF}(2)$ 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 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* $\mathcal{R}_{2^k}(d, m)$ of *order* d, where $d < m(2^k - 1)$, has length $N = 2^{km}$ over the field $F = \mathrm{GF}(2^k)$. 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 $\vec{a} = (a_1, \ldots, a_m)$ ranges over all sequences of arguments in F. In the important case $d \leq 2^k - 2$ 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 $d \leq 2^k - 2$ 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 is that for every two distinct polynomials f and g over F of total degree at most d, and for every $I \subseteq F$,

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

With $I = F$, it follows that the density Δ is at least $1 - d/|F|$. See [BFLS91, Sud92] for more on the inequality (1), and [MS77, TV91] for further information on the above codes. (Note: the notation in [TV91] for the generalized Reed-Muller code is $\mathcal{R}_q(r,m)$, where q is a prime power and $r < m(q-1)$. Below we will have $d = d_0 m$.)

3.2 Application for reductions to parity

Let $R(x,y)$ 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 δ. The *idea* is to find a $2^q \times 2^{r(n)}$ generator matrix G for a binary code C of constant density δ. 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 $R(x, y_i) \wedge G(i,j) = 1$.

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 GF(2). Then $\#acc(N_j, x)$ is odd iff the jth entry of w_S is a '1'. Since the proportion of non-0 entries of w_S is at least δ, $\Pr_j[\#acc(N_j,x) \text{ is odd}] \geq \delta$; that is, N reduces A to parity with success probability at least δ. And if S is empty, N has no accepting computations at all.

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

- Selected entries $G(i,j)$ are computable in quasilinear time, and
- The density δ of C is constant, the closer to 1/2 the better.

In one level of coding over 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 $\mathcal{R}_{2^k}(d,m)$, which has length N and density Δ over GF(2^k), may instead be regarded as a binary code $\mathcal{R}'$ of length kN over GF(2). But then we can only assert that the density of $\mathcal{R}'$ is at least Δ/k, because two distinct elements $a_1, a_2 \in$ 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 $\mathcal{H}_k$. Then whenever $a_1 \neq a_2$ in GF(2^k), $\mathcal{H}_k(a_1)$ and $\mathcal{H}_k(a_2)$ 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 $\mathrm{GF}(2^k)$ is prohibitive.

Let $\log^+ n$ abbreviate $\log n \operatorname{loglog} n \operatorname{logloglog} n$.

Theorem 3.1 *Let q be a quasilinear function. For every language A in* $\mathrm{NTIME}[q(n)]$, *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*

$$2q - q \operatorname{loglog} q / \log q + (1 + \log(1/\epsilon))q/\log q + O(\log q).$$

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

1. $n := |x|$, $q := q(n)$
2. $b := \lceil \log_2 q \rceil$ /*block length for exponents*/
3. $d_0 := 2^b - 1$ /*maximum degree in each variable*/
4. $m := \lceil q/b \rceil$ /*number of variables*/
5. $k := \lceil \log_2 d_0 + \log_2 m + \log_2(1/\epsilon) - 1 \rceil$
6. Calculate an irreducible polynomial α of degree k over $\mathrm{GF}(2)$
7. Flip $mk + k$ coins to form $j = \langle a_1, \ldots, a_m, v \rangle$, where $v \in \{0, 1\}^k$
8. Guess $y \in \{0, 1\}^q$
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 $R(x, y) \wedge G(y, j) = 1$.

Steps 1–5 take linear time. That step 6 can be done deterministically in time polynomial in k was shown by Shoup [Sho88], and since k is approximately $\log q + \log n + \log(1/\epsilon)$, which is $O(\log n)$ when ϵ is fixed, this time is negligible. Step 7 takes time about $nk/\log n$, which for fixed ϵ 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 $\mathrm{GF}(2)$ and reduce them modulo α in the field $\mathrm{GF}(2^k)$ takes time $t_1 = O(k \log k \operatorname{loglog} k)$ on standard Turing machine models (see [AHU74] and [Rab80]). The time to compute a^i in $\mathrm{GF}(2^k)$ where $i \leq q$ is $t_2 = O(\log q \cdot 2k \log k \operatorname{loglog} k)$ 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 $\mathrm{NQL} \subseteq \mathrm{RQL}[\oplus\mathrm{QL}]$. $\square$

The first open problem is whether two or more alternations can be done in quasilinear time; that is, whether $\mathrm{NQL}^{\mathrm{NQL}} \subseteq \mathrm{BQL}[\oplus\mathrm{QL}]$. 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 $R = K/N = 2^q/2^{(2q-\dots)}$, which tends to 0 as q increases. Families of codes are known for which R (as well as δ) 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 *prefix-set* method (cf. [Sel88]). To illustrate, let f be an arbitrary length-preserving, partial function from Σ^* to Σ^*. 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. Fix a universal Turing machine M_U for the definition of *Kolmogorov complexity* (see [LV93]).

Theorem 4.1 *There exist length-preserving functions $f : \Sigma^* \to \Sigma^*$ 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.*

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, a finite description of the function computing the time bound $g(n)$, and a finite description of the part of this proof that tells M_U how to assemble f from the given data. (See the use of the term "this discussion" in [LV93].) This has total length some constant C.
- A look-up table for all the strings of length less than n that 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 Kolmogorov complexity of f (relative to M_U). Then $C + 2^n + 2^n g(n)/n \geq K_f$, so $g(n) \geq nK_f/2^n - n - nC/2^n$. Since functions $f : \{0,1\}^n \to \{0,1\}^n$ 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: . Via diagonalization rather than Kolmogorov complexity, one can construct f so that it is computable in exponential time. 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.)

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.

Recall that a binary predicate R is called a *witness predicate* for a language L if $L = \{x : (\exists y)\, R(x,y)\}$. Associated to R is the *search problem*: given x, on condition $x \in L$, find some y such that $R(x,y)$ holds. We represent this problem by the partial multivalued function f_R defined for all x by:

$$f_R(x) \mapsto y, \text{if } R(x,y),$$

calling this the *search function* for L given by R. A solution to the search problem is a deterministic algorithm A such that for all $x \in L$, $A(x)$ outputs some y such that $f_R(x) \mapsto y$; i.e., such that $R(x,y)$ holds. The main branch-point of the theory concerns what A should do on inputs $x \notin L$, which is related to the complexity of checking $R(x,y)$. The general theory presented by Selman [Sel94] considers cases where R is not polynomial-time checkable but there is a solution A with the property $A(x) = 0$ for all $x \notin L$; and/or where A is nondeterministic but single-valued on inputs $x \in L$. Related matters are studied in [BD76, BBFG91, NOS93]. For our purposes, we decouple the runtime of A from the complexity of checking $R(x, y)$.

Definition 4.1. Let L be any language, and let t_1, t_2 be time-bound functions. Say that *search reduces to decision for L in time $t_1(n)$, with witness-checking in time $t_2(n)$*, if there exists a witness predicate R for L, a deterministic machine M_R computing $R(x, y)$, and a deterministic oracle machine M such that for all x:

(a) $M^L(x)$ halts within $t_1(|x|)$ steps, and for all y with $|y| \leq t_1(|x|)$, $M_R(x, y)$ halts within $t_2(|x|)$ steps.

(b) If $x \in L$, then $M^L(x)$ outputs some y such that $R(x, y)$ holds.

Note the existential quantification over all witness predicates R for L. In the case $t_2(n) = t_1(n) = t(n)$, we simply say that *search reduces to decision for L in time $t(n)$*, or *L has SRD in time $t(n)$* for short. An important subtlety is that this is stronger than stipulating that for all $x \notin L$, $M^L(x) = 0$. The ability to check $R(x, y)$ for all x, y makes it fruitful to simulate M on other oracles besides L itself. A final remark is that "linear speedup" does not hold in general for oracle machines; in particular, $t_1(n)$ not "$O(t_1(n))$" is the asserted bound on the total number of bits in queries made by M. In keeping with the promise in the Introduction we allow M to be an oracle version of any of the RAM-related models considered by Gurevich and Shelah, but arrange for our non-oracle simulator M' to be a standard multitape Turing machine.

Theorem 4.2 *Let search reduce to decision for L in time $t(n)$ with witness-checking in time $t(n)^{O(1)}$. Suppose that $t(\lfloor n/2 \rfloor) \leq t(n)/2$ for all but finitely many n. Then L is decidable by a non-oracle Turing machine in time $2^{2t(n)\log n/n} \cdot t(n)^{O(\log n)}$.*

Proof. Let M be the oracle machine from Definition 4.1 that runs in time $t(n)$, and let n_0 be a constant whose value will be fixed later. Also let $g(n)$ be a function that we subsequently fix as $g(n) = \lfloor n/2 \rfloor$ but retain in the first part of the analysis for later use. It is not important whether $t(n)$ is time-constructible.

We describe a non-oracle Turing machine M' that accepts L as follows: If the input x to M' has length less than n_0, then whether $x \in L$ is looked up in a table; this takes no more than n_0 steps. For inputs 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 $|z| > g(n)$, then M' simulates both a "yes" and a "no" answer to z. Finally, if $n_0 \leq |z| \leq g(n)$, then M' calls itself recursively on input z to answer the query.

In greater machine detail: The TM M' has one special "stack tape" as well as its own worktapes. One worktape represents the query tape of M', and another acts as a surrogate input tape in recursive calls. For segments of computations by M between queries, M' simulates M on its worktapes using any of the standard on-line simulations of RAM-like models by TMs, which carry quadratic overheads in time and space (see [CR73, PF79, WW86]). When M' encounters a query z with $|z| > g(n)$, M' pushes a copy of its current configuration *aside from the stack* onto the stack, and continues the simulation from that *branch point* down the '0' branch. This configuration has size at most $t(n)^2$. For a query z with $n_0 < |z| \leq g(n)$, M' again pushes a copy of its current configuration onto the stack, copies z to its surrogate input tape, and begins *working on* z. Pushing onto the stack takes no more than $t(n)^2$ steps in this case also. The value n_0 is global, but the value $g(|z|)$ is recomputed at each level of the recursion.

The main point of the simulation is that both the branchings and the recursive calls can be handled without conflicts on the single stack. At all stages there is a current string z, $|z| \geq n_0$, that M' is working on. At the bottom of the recursion, M' has simulated some branch of the oracle computation of $M^{(\cdot)}(z)$ to its completion. If the branch returns a value y, then M' itself tests $R(z, y)$. If $R(z, y)$ holds, then M' pops its stack until it finds the configuration that queried z, and proceeds from there with a "yes" answer. If the branch does not return such a y, or returns 0, then M' retrieves the last branch point from the stack and simulates the '1' branch from that point. If there are no more branch points left; i.e., if the configuration that queried z is uppermost, then M' proceeds from there with a "no" answer. The whole computation is a left-to-right transversal of the "hierarchical tree" formed by the branch points and recursive calls; we picture this intuitively as a "flattened," wide-spreading tree.

Let $T(n)$ stand for the worst-case run-time of M' on inputs of length n. Let $t_R(n)$ stand for the time to decide R. Then for all $n < n_0$, $T(n) \leq n_0$, while for $n \geq n_0$, $T(n)$ satisfies the bound

$$T(n) \leq 2^{t(n)/g(n)} \cdot t(n)(t(n)^2 + T(g(n)) + 2^{t(n)/g(n)} t_R(n). \tag{2}$$

To see this, first note that there can be at most $t(n)$ (actually, $t(n)/n_0$) recursive calls along any branch of the simulation of $M^{(\cdot)}(x)$. Each call involves pushing a configuration of size at most $t(n)^2$ onto the stack, and in the same time copying z to the surrogate-input tape and re-setting the other worktapes of M'. Since each branch is for a query of length at least $g(n)$, there can be at most $2^{t(n)/g(n)}$ such branches.

Using the hypothesis that $t_R(n) = t(n)^{O(1)}$, and letting c be the constant in the "$O(1)$" plus 3, we obtain the even cruder upper bound

$$T(n) \leq 2^{t(n)/g(n)} \cdot t(n)^c \cdot T(g(n)). \tag{3}$$

Now let $g(n) := \lfloor n/2 \rfloor$ for all n. Fix n_0 so that $t(\lfloor n/2 \rfloor) \leq t(n)/2$ for all $n \geq n_0$. Put $k = \lceil \log_2(n/n_0) \rceil$. Unwinding the recursion, we obtain for $n \geq n_0$:

$$
\begin{aligned}
T(n) &\leq 2^{\sum_{i=0}^{k} t(n/2^i)/(n/2^{i+1})} \cdot t(n)^{ck} \\
&\leq 2^{2t(n)\log n/n} \cdot t(n)^{c \log n}.
\end{aligned}
$$

$\square$

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

Corollary 4.3 *If search reduces to decision for SAT in quasilinear time with polynomial witness-checking, then* NP $\subseteq$ QP.

The same conclusion holds under the hypothesis that some (any) NP-complete language L has SRD in quasilinear time, with polynomial witness-checking.

Proof. For quasilinear $t(n)$, $t(n)/n = \mathrm{polylog}(n)$, and since $n^{\mathrm{polylog}(n)} = 2^{\mathrm{polylog}(n)}$, SAT would belong to DTIME$[2^{\mathrm{polylog}(n)}]$. Since $\mathrm{polylog}(n^c) = \mathrm{polylog}(n)$, the conclusion extends to all of NP. $\square$

In fact, letting NQP stand for nondeterministic quasi-polynomial time, we would have the conclusion NQP = QP. The next result follows by the same token, except that the nearly-2^{n^ϵ} time bound need not extend to all of NP, just to NQL.

Corollary 4.4 *If there exists an $\epsilon > 0$ such that SAT has SRD in time $O(n^{1+\epsilon})$, with polynomial witness-checking, then for all $\delta > \epsilon$,* NQL $\subseteq$ DTIME$[2^{n^\delta}]$. $\square$

Stearns and Hunt [SH90] define a language $L \in$ NP to have *power index* ϵ if ϵ is the infimum of all δ such that $L \in$ 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 time $O(n^{1+\epsilon})$, then SAT has power index at most ϵ.* $\square$

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 DTIME$[2^{2^{O(n)}}]$, 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. $\square$

Now we push these results right up against the quadratic upper bound of Theorem 4.1 (which applies when $|y| \approx |x|$), needing to use only one level of recursion. Say that a function $t(n)$ is "recursively $o(n^2)$" if there is a total recursive function $f : \mathbf{N} \to \mathbf{N}$ such that $f(n) \to \infty$ as $n \to \infty$ and for all but finitely many n, $t(n) \leq n^2/f(n)$.

Theorem 4.7 *If SAT has SRD in time $t(n)$ that is recursively $o(n^2)$, with witness-checking in polynomial time, then $SAT \in$* DTIME$[2^{o(n)}]$.

Proof. Given the computable function f associated to $t(n)$ in the preceding remarks, we replace f by an easily computable lower bound that still goes off to ∞ by the following standard trick: For all n, define $f'(n)$ to be the largets value obtained by simulating the computation of $f(0), f(1), f(2), \ldots$ for n steps. Now for all (sufficiently large) n define $g(n) := \lfloor n/\sqrt{f'(n)} \rfloor$. The time to compute $g(n)$ is polynomial in n, and absorbing this into the polynomial bound for witness-checking gives via (3), for some $c > 0$ and all sufficiently large n:

$$T(n) \leq 2^{n/\sqrt{f'(n)}} \cdot n^c \cdot T(n/\sqrt{f'(n)}).$$

Now instead of continuing the recursion below the top level as in Theorem 4.2, let M' on each query string z with $|z| \leq g(n)$ run the brute-force $2^{O(|z|)}$ time algorithm for SAT instead. Then the running time of M' is bounded by $n^c \cdot 2^{O(n/\sqrt{f'(n)})} = 2^{o(n)}$. $\square$

The result extends to show that if the time to reduce search to decision for SAT is not $\Omega(n^2)$, then the deterministic time complexity of SAT is not $\Omega(2^n)$ or even $2^{\Omega(n)}$. One can also analyze the full recursion without resorting to the membership of SAT in DTIME$[2^{O(n)}]$ (which is the same for TMs and RAMs), obtaining essentially the same kind of bound. Analogous results hold for general languages in NP in place of SAT.

Finally, a remark on the nature of the recursion in Theorem 4.2: It is natural to wonder whether this equivalent to the brute-force simulation of a machine with "limited nondeterminism" in the sense of [KF80, BG93]. That is, perhaps L is accepted by a polynomial-time NTM that makes at most $t(n)/n$ or so nondeterministic moves on inputs of length n; this would be polylog(n) or n^{ϵ}-limited nondeterminism in the cases of Corollaries 4.3 and 4.4. The answer appears to be no, owing to the fact that "no" answers to queries $z \notin L$ are relied upon in the simulation. We suspect that some restricted notion of "search reducing to decision by positive queries alone," analogous to the notion of *positive* Turing reducibility (see [Sel82, Sch85a]), is needed for such a connection to limited nondeterminism. The relativized search-to-decision reduction used in the main theorem of the next section is positive in the appropriate sense.

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 [Sch85b] 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 $L(M^B) = L(M^{\emptyset})$. 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 $L(M^{(\cdot)}) = A$ 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* $O(t(n))$ if A 1-sided helps A itself in time $O(t(n))$.

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

Proof. ($\Longrightarrow$): 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 $f_R(x) \mapsto y$, 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 $O(t(n))$.

($\Longleftarrow$): 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 $R(x, y) \equiv$ "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 $R \in \mathrm{DTIME}[t(n)]$. 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 $2t(n)$. $\square$

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

We can, however, prove a more general theorem in terms of 1-sided-helping, without the "self-" restriction. For some notation, let $\mathrm{P}_{\text{1-help}}(\mathcal{C})$ and $\mathrm{DQL}_{\text{1-help}}(\mathcal{C})$ denote, respectively, the classes of languages A such that there is a language $B \in \mathcal{C}$ which 1-sided-helps A in polynomial time, respectively, in quasilinear time. Ko proved that for *any* complexity class $\mathcal{C}$ that contains an NP-hard language, $\mathrm{NP} = \mathrm{P}_{\text{1-help}}(\mathcal{C})$ [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 $\mathcal{C}$, $\mathrm{DQL}_{\text{1-help}}(\mathcal{C}) \subseteq \mathrm{NQL}$.*

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

Proof.

(a) Let $L \in \mathrm{DQL}_{\text{1-help}}(A)$ 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 $L \in \mathrm{NQL}$, 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 $x \notin L$, 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 $SAT \in \mathrm{DQL}_{\text{1-help}}(\mathrm{NQL})$. Then there exists some $A \in \mathrm{NQL}$ that helps SAT via a robust OTM M in quasilinear time. Since $A \leq_m^{ql} SAT$, 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, BGGO93]. In particular, a language L is defined to be P-*superterse* [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-superterse iff $\mathrm{P} \neq \mathrm{NP}$. Their proof relativizes in this way: for any oracle A such that $\mathrm{P}^A \neq \mathrm{NP}^A$, 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 SAT^A with oracle $0A \cup 1B$ must make at least k queries to the B half of the oracle. Since superterseness 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 $\mathrm{P}^A \neq \mathrm{NP}^A$, 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. Our oracle construction is related to those of Kintala and Fischer [KF80] for polynomial-time NTMs allowed to make $O(\log^k n)$ nondeterministic moves, for some $k > 0$. 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 $L_1 \equiv_m^{ql} L_2$ 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 $L_1 \leq_m^{ql} L_2$ via g_1 and $L_2 \leq_m^{ql} L_1$ 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 $\{Q_i\}_{i\in N}$ be an enumeration of nondeterministic quasilinear time Turing machines. Then for any oracle X, the following standard language is complete for NQL^X under $\leq_m^{ql}$ reductions:

$$K^X = \{\, \langle x, i, y\rangle : Q_i \text{ with oracle } X \text{ accepts } x \text{ within } |y| \text{ steps } \}.$$

For convenience, we modify K^X into another NQL-complete language $padK^X$ such that the lengths of all strings in $padK^X$ are powers of 2:

$$padK^X = \{\, \langle x, i, y, 0^r\rangle : r < |\langle x, i, y\rangle|,\ |\langle x, i, y, 0^r\rangle| \text{ is a power of 2, and } \langle x, i, y\rangle \in K^X \,\}.$$

It is clear that K^X reduces to $padK^X$ in linear time, so that $padK^X$ 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) $L^A \in \mathrm{NP}^A - \mathrm{P}^A$, and

(ii) for all $u \in \Sigma^*$, $u \in padK^A$ 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 $\sim A$, and some strings may be added to A. If nothing is done at stage n, then all strings of length n belong to $\sim A$. No decisions on membership are ever changed, and no requirements are "injured."

Now let $\{M_i\}_{i\in N}$ 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 $padK^A$, we focus on lengths ℓ that are a power of 2.

Stage $n = 2m$**:** If there does not exist a power-of-2 ℓ 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 $\sim A$ at an earlier stage, determine the unique string x of length ℓ that is a prefix of z. Add z to A if and only if x is in $padK^{A(n)}$.

Stage $n = 2m + 1$**:** If there exists a string of length $\geq n$ that has been reserved for $\sim A$ 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 ℓ 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 $\sim A$ 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 $L^A \in \mathrm{NP}^A - \mathrm{P}^A$.

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 ℓ_n such that $n + 1 = e(\ell_n)$. Hence for infinitely many odd n,

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

Since no action at even stages reserves any string for $\sim A$, 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,*

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

Proof. It suffices to show that for all powers-of-2 ℓ and all strings $x \in \Sigma^\ell$, there exists a string $v \in \Sigma^{\log^2 \ell}$ such that xv is never reserved for $\sim A$. 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 ℓ_i such that $e(\ell_i) = n_i + 1$, and also $\ell_i \leq \ell$. Since n_i was the last canceling stage, and $n_i < e(\ell)$, any string of length $e(\ell)$ that was reserved for $\sim A$ before step $e(\ell)$ was reserved at stage n_i. From the fact that i was canceled, $p_i(n_i) < 2^{\log^2 l_i}$. Hence the number of strings of length $e(\ell)$ that were reserved for $\sim A$ 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 $p_i(n_i) < 2^{\log^2 \ell_i} \leq 2^{\log^2 \ell}$. Since there are $2^{\log^2 \ell}$-many strings v such that $xv \in \Sigma^{e(\ell)}$, at least one such string is never reserved for $\sim A$. 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 language in NQL^A:

$$\mathit{prefixK}^A = \{ \langle x, v\rangle : (\exists u \in A)\, [|u| = e(|x|) \text{ and } xv \text{ is a prefix of u} \}.$$

By the construction of A, it follows that padK^A reduces in linear time to $\mathit{prefixK}^A$, and since padK^A is NQL^A-complete, $\mathit{prefixK}^A$ is also NQL^A-complete. Since witnesses to membership of a string x in $\mathit{prefixK}^A$ 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 $\mathit{prefixK}^A \equiv_m^{ql} \mathit{padK}^A \equiv_m^{ql} \mathit{SAT}^A$. $\square$

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 $\mathrm{P} = \mathrm{NP}$ can be shown to follow if 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 NP-complete sets, and 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) SAT has power index 1 [SH86].

(b) SAT is P-superterse [Bei87a].

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

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

(e) The search function for 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 $\mathrm{PF}^{\mathrm{NP}[O(\log n)]}$ unless P = NP. There has been considerable interest in whether these functions can be shown to be outside $\mathrm{PF}^{\mathrm{NP}[o(n)]}$ or even $\mathrm{PF}^{\mathrm{NP}[O(\log^2 n)]}$ unless P = NP. Theorem 4.2 provides a viewpoint on this question: if the largest clique can be found in $\mathrm{PF}^{\mathrm{NP}}$ using at most $n \operatorname{polylog} n$ query bits, then NP $\subseteq$ DTIME$[2^{\operatorname{polylog} n}]$, and $n^{1+\epsilon}$ query bits would place MaxClique into DTIME$[2^{n^\epsilon}]$. The closest impact of (e) may be in relation to (d). By results of Juedes and Lutz [JL93], (d) implies that there exists $\epsilon > 0$ such that SAT does not have power index ϵ, hence that search does not reduce to decision for SAT in time $O(n^{1+\epsilon})$. 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 $k \geq 1$, 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 $\neq$ UQL, then QL one-way functions exist, but unlike the polynomial case (assuming P $\neq$ 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 $\mathrm{DQL}^A = \mathrm{NQL}^A$. 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. We also thank the anonymous referees for catching a number of technical glitches in our earlier draft, and for helpful suggestions.

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 Structure in Complexity Theory*, pages 232–243, 1990.

[ABN+92] N. Alon, J. Bruck, J. Naor, M. Naor, and R. Roth. Construction of asymptotically good low-rate error-correcting codes through pseudo-random graphs. *IEEE Transactions on Information Theory*, 38(2):509–512, March 1992.

[AG88] A. Amir and W. Gasarch. Polynomial terse sets. *Inform. and Control*, 77:37–56, 1988.

[AHU74] A. Aho, J. Hopcroft, and J. Ullman. *The Design and Analysis of Computer Algorithms*. Addison-Wesley, Reading, Mass., 1974.

[Bal90] J. Balcázar. Self-reducibility structures and solutions of NP problems. *Rivesta Matematica*, 4(4):175–184, Dec. 1990.

[Bar92] D. Mix Barrington. Some problems involving Razborov-Smolensky polynomials. In M. Paterson, editor, *Boolean Function Complexity*, volume 169 of *LMS Lecture Note Series*, pages 109–128. London Math. Soc., 1992. Proceedings of an LMS Symposium in Durham, July 1990.

[BBFG91] R. Beigel, M. Bellare, J. Feigenbaum, and S. Goldwasser. Languages that are easier than their proofs. In *Proc. 32nd Annual IEEE Symposium on Foundations of Computer Science*, pages 19–28, 1991.

[BCGL89] S. Ben-David, B. Chor, O. Goldreich, and M. Luby. On the theory of average-case complexity. In *Proc. 21st Annual ACM Symposium on the Theory of Computing*, pages 204–216, 1989.

[BD76] A. Borodin and A. Demers. Some comments on functional self-reducibility and the NP hierarchy. Technical Report TR 76-284, Cornell Univ. Comp. Sci. Dept., 1976.

[BDG88] J. Balcázar, J. Díaz, and J. Gabarró. *Structural Complexity I*. Springer Verlag, 1988.

[Bei87a] R. Beigel. *Query-limited reducibilities*. PhD thesis, Stanford University, 1987.

[Bei87b] R. Beigel. A structural theorem that depends quantitatively on the complexity of SAT. In *Proc. 2nd Annual IEEE Conference on Structure in Complexity Theory*, pages 28–32, 1987.

[Bei91] R. Beigel. Bounded queries to SAT and the Boolean hierarchy. *Theor. Comp. Sci.*, 83:199–223, 1991.

[BFLS91] L. Babai, L. Fortnow, L. Levin, and M. Szegedy. Checking computations in polylogarithmic time. In *Proc. 23rd Annual ACM Symposium on the Theory of Computing*, pages 21–31, 1991.

[BG93] J. Buss and J. Goldsmith. Nondeterminism within P. *SIAM J. Comput.*, 22:560–572, 1993.

[BG94] R. Beigel and J. Goldsmith. Downward separation fails catastrophically for limited nondeterminism classes. In *Proc. 9th Annual IEEE Conference on Structure in Complexity Theory*, pages 134–138, 1994.

[BGGO93] R. Beigel, W. Gasarch, J. Gill, and C. Owings. Terse, superterse, and verbose sets. *Inform. and Comp.*, 103:68–85, 1993.

[BGH89] R. Beigel, W. Gasarch, and L. Hay. Bounded query classes and the difference hierarchy. *Archive for Mathematical Logic*, 29:69–84, 1989.

[BKS93] R. Beigel, M. Kummer, and F. Stephan. Approximable sets. manuscript, 1993.

[Coo71] S. Cook. The complexity of theorem-proving procedures. In *Proc. 3rd Annual ACM Symposium on the Theory of Computing*, pages 151–158, 1971.

[CR73] S. Cook and R. Reckhow. Time bounded random access machines. *J. Comp. Sys. Sci.*, 7:354–375, 1973.

[CRS93] S. Chari, P. Rohatgi, and A. Srinivasan. Randomness-optimal unique element isolation, with applications to perfect matching and related problems. In *Proc. 25th Annual ACM Symposium on the Theory of Computing*, pages 458–467, 1993.

[CW79] J. Carter and M. Wegman. Universal classes of hash functions. *J. Comp. Sys. Sci.*, 18:143–154, 1979.

[For66] G. Forney. *Concatenated Codes*. MIT Press, 1966.

[GJ79] M. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman, 1979.

[Gra90] E. Graedel. On the notion of linear-time computability. *Intern. J. Found. Comp. Sci.*, 1:295–307, 1990.

[GS89] Y. Gurevich and S. Shelah. Nearly-linear time. In *Proceedings, Logic at Botik '89*, volume 363 of *Lect. Notes in Comp. Sci.*, pages 108–118. Springer Verlag, 1989.

[Gup93] S. Gupta. On isolating an odd number of elements and its applications to complexity theory. Technical Report OSU-CISRC-6/93-TR24, Dept. of Comp. Sci., Ohio State University, 1993.

[HN93] A. Hoene and A. Nickelsen. Counting, selecting, and sorting by query-bounded machines. In *Proc. 10th Annual Symposium on Theoretical Aspects of Computer Science*, volume 665 of *Lect. Notes in Comp. Sci.*, pages 196–205. Springer Verlag, 1993.

[HU79] J. Hopcroft and J. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison–Wesley, Reading, MA, 1979.

[HW89] S. Homer and J. Wang. Absolute results concerning one-way functions and their applications. *Math. Sys. Thy.*, 22:21–35, 1989.

[JL93] D. Juedes and J. Lutz. The complexity and distribution of hard problems. In *Proc. 34th Annual IEEE Symposium on Foundations of Computer Science*, pages 177–185, 1993. SIAM J. Comput., to appear.

[JLJH92] J. Justesen, K. Larsen, H.E. Jensen, and T. Hoholdt. Fast decoding of codes from algebraic plane curves. *IEEE Transactions on Information Theory*, 38(1):111–119, January 1992.

[JY90] D. Joseph and P. Young. Self-reducibility: the effects of internal structure on computational complexity. In A. Selman, editor, *Complexity Theory Retrospective*, pages 82–107. Springer Verlag, 1990.

[KF80] C. Kintala and P. Fischer. Refining nondeterminism in relativized polynomial-time bounded computations. *SIAM J. Comput.*, 9:46–53, 1980.

[Ko82] K. Ko. Some observations on the probabilistic algorithms and NP-hard problems. *Inf. Proc. Lett.*, 14:39–43, 1982.

[Ko87] K. Ko. On helping by robust oracle machines. *Theor. Comp. Sci.*, 52:15–36, 1987.

[Kre88] M. Krentel. The complexity of optimization problems. *J. Comp. Sys. Sci.*, 36:490–509, 1988.

[LL76] R. Ladner and N. Lynch. Relativization of questions about log-space computability. *Math. Sys. Thy.*, 10:19–32, 1976.

[Lut93] J. Lutz. The quantitative structure of exponential time. In *Proc. 8th Annual IEEE Conference on Structure in Complexity Theory*, pages 158–175, 1993.

[LV93] M. Li and P. Vitányi. *An Introduction to Kolmogorov Complexity and its Applications*. Springer Verlag, 1993.

[MS77] F. MacWilliams and N. Sloane. *The Theory of Error-Correcting Codes*. North-Holland, Amsterdam, 1977.

[NN90] J. Naor and M. Naor. Small-bias probability spaces: efficient constructions and applications. In *Proc. 22nd Annual ACM Symposium on the Theory of Computing*, pages 213–223, 1990.

[NN93] J. Naor and M. Naor. Small-bias probability spaces: efficient constructions and applications. *SIAM J. Comput.*, 22:838–856, 1993.

[NOS93] A. Naik, M. Ogiwara, and A. Selman. P-selective sets, and reducing search to decision vs. self-reducibility. In *Proc. 8th Annual IEEE Conference on Structure in Complexity Theory*, pages 52–64, 1993.

[Pap94] C. Papadimitriou. *Computational Complexity*. Addison-Wesley, Reading, Mass., 1994.

[PF79] N. Pippenger and M. Fischer. Relations among complexity measures. *J. Assn. Comp. Mach.*, 26:361–381, 1979.

[PZ83] C. H. Papadimitriou and S. Zachos. Two remarks on the power of counting. In *The 6th GI Conference on Theoretical Computer Science*, Lecture Notes in Computer Science No. 145, pages 269–276. Springer Verlag, 1983.

[Rab80] M. Rabin. Probabilistic algorithms in finite fields. *SIAM J. Comput.*, 9:273–280, 1980.

[Sch76] C. Schnorr. The network complexity and the Turing machine complexity of finite functions. *Acta Informatica*, 7:95–107, 1976.

[Sch78] C. Schnorr. Satisfiability is quasilinear complete in NQL. *J. Assn. Comp. Mach.*, 25:136–145, 1978.

[Sch85a] U. Schöning. *Complexity and Structure*, volume 211 of *Lect. Notes in Comp. Sci.* Springer-Verlag, 1985.

[Sch85b] U. Schöning. Robust algorithms: a different approach to oracles. *Theor. Comp. Sci.*, 40:57–66, 1985.

[Sel79] A. Selman. P-selective sets, tally languages, and the behavior of polynomial time reducibilities on NP. *Math. Sys. Thy.*, 13:55–65, 1979.

[Sel82] A. Selman. Reductions on NP and p-selective sets. *Theor. Comp. Sci.*, 19:287–304, 1982.

[Sel88] A. Selman. Natural self-reducible sets. *SIAM J. Comput.*, 17:989–996, 1988.

[Sel94] A. Selman. A taxonomy of complexity classes of functions. *J. Comp. Sys. Sci.*, 48:357–381, 1994.

[SH86] R. Stearns and H. Hunt III. On the complexity of the satisfiability problem and the structure of NP. Technical Report 86–21, Dept. of Comp. Sci., SUNY at Albany, 1986.

[SH90] R. Stearns and H. Hunt III. Power indices and easier hard problems. *Math. Sys. Thy.*, 23:209–225, 1990.

[She93] B.-Z. Shen. A Justesen construction of binary concatenated codes than asymptotically meet the Zyablov bound for low rate. *IEEE Transactions on Information Theory*, 39(1):239–242, January 1993.

[Sho88] V. Shoup. New algorithms for finding irreducible polynomials over finite fields. In *Proc. 29th Annual IEEE Symposium on Foundations of Computer Science*, pages 283–290, 1988.

[Sto76] L. Stockmeyer. The polynomial time hierarchy. *Theor. Comp. Sci.*, 3:1–22, 1976.

[Sud92] M. Sudan. *Efficient checking of polynomials and proofs and the hardness of approximation problems.* PhD thesis, University of California, Berkeley, 1992.

[Tod91] S. Toda. PP is as hard as the polynomial-time hierarchy. *SIAM J. Comput.*, 20:865–877, 1991.

[Tre86] C. Tretkoff. Bounded oracles and complexity classes inside linear space. In *Proc. 1st Annual IEEE Conference on Structure in Complexity Theory*, volume 223 of *Lect. Notes in Comp. Sci.*, pages 347–361. Springer Verlag, 1986.

[TV91] M. Tsfasman and S. Vladut. *Algebraic-Geometric Codes*, volume 58 of *Mathematics and Its Applications (Soviet Series)*. Kluwer Academic, Dordrecht, 1991.

[vEB90] P. van Emde Boas. Machine models and simulations. In J. Van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume A, pages 1–66. Elsevier and MIT Press, 1990.

[VV86] L. Valiant and V. Vazirani. NP is as easy as detecting unique solutions. *Theor. Comp. Sci.*, 47:85–93, 1986.

[vzG91] J. von zur Gathen. Efficient exponentiation in finite fields. In *Proc. 32nd Annual IEEE Symposium on Foundations of Computer Science*, pages 384–391, 1991.

[Wra76] C. Wrathall. Complete sets and the polynomial-time hierarchy. *Theor. Comp. Sci.*, 3:23–33, 1976.

[Wra78] C. Wrathall. Rudimentary predicates and relative computation. *SIAM J. Comput.*, 7:194–209, 1978.

[WW86] K. Wagner and G. Wechsung. *Computational Complexity.* D. Reidel, 1986.