JOURNAL OF COMPLEXITY 1, 210–231 (1985)

Continuous Optimization Problems and a Polynomial Hierarchy of Real Functions*,†

KER-I KO

Department of Computer Science, University of Houston, Houston, Texas 77004

The close connection between the maximization operation and nondeterministic computation has been observed in many different forms. We examine this relationship on real functions and give a characterization of NP-time computable real functions by the maximization operation. A natural extension of NP-time computable real functions to a polynomial hierarchy of real functions has a characterization by alternating operations of maximization and minimization. Although syntactically this hierarchy of real functions can be treated as a polynomial hierarchy of operators, the well-known Baker–Gill–Solovay separation result does not apply to this hierarchy. This phenomenon is explained by the inherent structural properties of real functions, and is compared with recent studies on positive relativization.

1. INTRODUCTION

In the study of the theory of NP-completeness, a great number of natural combinatorial optimization problems have been shown to be NP-complete. (See Garey and Johnson (1979) for a complete list.) Since all these optimization problems have the same computational complexity and similar structures, it is natural to try to study these problems uniformly in a general framework. For example, for the purpose of a theory of approximation algorithms, Johnson (1974) has given a general formulation of a class of optimization problems:

> Given (a) a set D of problem instances, (b) for each instance I in D a finite set $S(I)$ of candidate solutions for I, and (c) for each solution $\sigma \in S(I)$ a positive rational number $m(I, \sigma)$, find for each

* Presented at the Symposium on Complexity of Approximately Solved Problems, April 17, 1985.

† This research was supported in part by National Science Foundation Grants MCS-8103479 and MCS-8103479A01.

0885-064X/85 $3.00

I in D the optimal solution $\sigma^* \in S(I)$ such that $m(I, \sigma^*)$ is maximized (or, minimized).

Often the size of the set $S(I)$ of solutions grows in an exponential rate as a function of the size of the instance I. Thus an exhaustive search algorithm would take time exponential in its input size. A good approximation algorithm should run in time polynomial in its input size and output a solution $\sigma \in S(I)$ with the value $m(I, \sigma)$ close to the optimal value $m(I, \sigma^*)$.

A more abstract formulation for the general maximization problem is the following:

> For a given polynomial-time computable function $f: N \to N$ and a given integer $x \in N$, find $\max \{f(y): y \leq x\}$.

More precisely, we ask whether the function $\max_f$, defined by $\max_f(x) = \max\{f(y): y \leq x\}$, is polynomial-time computable, provided that f is known to be polynomial-time computable. From the study of NP-complete combinatorial optimization problems, we naturally expect that the above question has an affirmative answer iff P = NP. Indeed, this is the case, as proved by Friedman (1984).

In this paper, we are concerned with the computational complexity of the maximization problem on real-valued functions. From the above discussion, it is natural to try to apply the concept of nondeterministic computation to this problem. In order to do so, however, we must define the continuous maximization problem in a discrete form. We follow the general approach of recursive analysis and the complexity theory of real functions developed by Ko and Friedman (1982). In particular, we consider a real number as a sequence of rational numbers that satisfies some convergence requirement, and a real function as an operator (or, a type 2 function) that maps rational number sequences to rational number sequences. In this model, the maximization problem on real functions is not even a type 2 function but a higher-order operator that maps type 2 functions (real functions) to type 1 functions (real numbers).

To illustrate how in general we may attack a problem involving with the computational complexity of a type 2 function, and to further demonstrate the close relationship between nondeterminism and maximization, let us consider the following question.

> Let MAX be the operator that maps each (integer) function f to the function $\max_f$. What is the computational complexity of MAX?

Before we try to apply the theory of NP-completeness to this problem, we note that the concept of NP computation is more natural when applied to language recognition problems, and so we first need to convert this operator

to a *set operator* that maps functions to sets. We define the operator PMAX as follows:

$$\text{PMAX}(f) = \{\langle x, y\rangle : y \leq \max\{f(z) : z \leq x\}\}.$$

Note that on polynomially length-bounded functions f, PMAX(f) is computable in polynomial time by an oracle Turing machine (oracle TM) iff MAX(f) is computable in polynomial time by an oracle TM.

Now we define P_{op} (NP_{op}) to be the class of all set operators F for which there is a deterministic (nondeterministic, respectively) polynomial-time oracle TM M such that for any function oracle f and input x, M^f accepts x iff $x \in F(f)$. Baker, Gill, and Solovay (1975) have shown that $\text{P}_{\text{op}} \neq \text{NP}_{\text{op}}$. Similar proof techniques can easily be applied to PMAX and show that $\text{PMAX} \in \text{NP}_{\text{op}} - \text{P}_{\text{op}}$. Actually, Ko (1985) has shown that PMAX is *complete* for the class NP_{op} in the following sense:

(i) $\text{PMAX} \in \text{NP}_{\text{op}}$, and

(ii) for every set operator $F \in \text{NP}_{\text{op}}$, there exists a polynomial-time oracle TM M such that for each function $f \in \text{domain}(F)$, M^f computes a function g and $F(f)$ is polynomial-time m-reducible (i.e., Karp-reducible) to PMAX(g).

We note that the result that PMAX is complete for NP_{op} implies more than $\text{PMAX} \notin \text{P}_{\text{op}}$. It actually implies that PMAX is not computable in polynomial time by any types of oracle machines which are provably weaker than a nondeterministic oracle TM; for example, a polynomial-time probabilistic oracle TM with one-sided error, or a polynomial-time unambiguous nondeterministic oracle TM. (A nondeterministic oracle TM is *unambiguous* if for every oracle and every input it accepts, there is exactly one accepting computation (Ko, 1985).)

The above approach to the study of computational complexity of type 2 functions suggests that we use oracle machines as a computational model for real functions and that we apply the concept of completeness to the class of real functions computable by polynomial-time oracle TMs. It should be pointed out, however, that a computable real function must be continuous, and this fact makes the completeness result difficult to prove, if possible at all. Instead, our main theorem (Theorem 6), which generalizes earlier results of Ko (1982) and Friedman (1984), is of a different form:

> A real function f defined on $[0, 1]$ is computable in polynomial time by a nondeterministic oracle TM iff there is a real function g, defined on $[0, 1]^2$, that is computable in polynomial time by a deterministic oracle TM, such that for all $x \in [0, 1]$, $f(x) = \max\{g(x, y): y \in [0, 1]\}$.

In other words, the operation of maximization characterizes the concept of NP-time real function computation. In Section 5, we show, in contrast to Baker, Gill, and Solovay's (1975) result, that the existence of a NP-time computable real function that is not polynomial-time computable is equivalent to P $\neq$ NP. This result shows that the continuity property affects strongly the computational complexity of type 2 functions.

Another interesting related question is the computational complexity of the alternating operations of maximization and minimization on real functions. Consider first the case of type 2 functions without the continuity constraint. For each $k \geq 1$, define PMAX_k to be the set operator such that for each function f,

$$\mathrm{PMAX}_k(f) = \{\langle x, z\rangle : z \leq \max_{y_1 \leq x} \min_{y_2 \leq x} \cdots \operatorname{opt}_{y_n \leq x} f(\langle y_1, \ldots, y_n\rangle)\},$$

where $\operatorname{opt}_k = \max$ if k is odd ($= \min$ if k is even). What is the computational complexity of PMAX_k? Anyone who is familiar with the Meyer–Stockmeyer polynomial-time hierarchy (Stockmeyer, 1976) would have guessed that the complexity of the operator PMAX_k is best classified by establishing a hierarchy of operators based on the number of alternating quantifiers. Indeed, we can extend in a natural way the Meyer–Stockmeyer polynomial-time hierarchy to a polynomial-time hierarchy of operators. For example, we say that $F \in \Sigma^{\mathrm{P}}_{k,\mathrm{op}}$ for some $k \geq 1$, if there is an operator G in P_{op}, and a polynomial function p, such that for all $f \in$ domain(F) and all x,

$$x \in F(f) \Leftrightarrow (\exists y_1, |y_1| \leq p(|x|))\,(\forall y_2, |y_2| \leq p(|x|)) \ldots (Q_k y_k, |y_k| \leq p(|x|))\,\langle x, y_1, \ldots, y_k\rangle \in G(f),$$

where $Q_k = \exists$ if k is odd ($= \forall$ if k is even). Then, by extending naturally the concept of completeness to the class $\Sigma^{\mathrm{P}}_{k,\mathrm{op}}$, we can prove that, for each $k \geq 1$, PMAX_k is complete for the class $\Sigma^{\mathrm{P}}_{k,\mathrm{op}}$ (Ko, 1985).

An immediate consequence of the above result and Baker and Selman's (1979) result that separates $\Sigma^{\mathrm{P}}_{2,\mathrm{op}}$ from $\Pi^{\mathrm{P}}_{2,\mathrm{op}}$ is that $\mathrm{PMAX}_2 \in \Pi^{\mathrm{P}}_{2,\mathrm{op}}$. Since it is not known at the present time whether $\Sigma^{\mathrm{P}}_{3,\mathrm{op}} = \Pi^{\mathrm{P}}_{3,\mathrm{op}}$ or not, we do not know whether PMAX_3 is in $\Pi^{\mathrm{P}}_{3,\mathrm{op}}$ or not. However, it implies that if any operator can distinguish $\Sigma^{\mathrm{P}}_{3,\mathrm{op}}$ from $\Pi^{\mathrm{P}}_{3,\mathrm{op}}$, so can PMAX_3.

For real functions, a similar hierarchy of real functions based on the number of alternating operations of maximization and minimization can be constructed (see Section 4). Similar to the Meyer–Stockmeyer polynomial hierarchy, this hierarchy of real functions is not known to be an infinite hierarchy. In fact, we can show that this hierarchy does not collapse iff the Meyer–Stockmeyer polynomial hierarchy does not collapse. This result is closely related to the recent results on "positive relativization" (Long and Selman, 1983) and will be discussed in Section 5.

2. P-Time and NP-Time Computable Real Functions

A theory of computational complexity of real functions has been developed in Ko and Friedman (1982). In this section, we give only the definitions and basic properties that are necessary to develop our results. The reader is referred to Ko and Friedman (1982) and Ko (1984) for a detailed discussion.

We assume the reader is familiar with Turing machines (TMs), oracle Turing machines (oracle TMs), nondeterministic Turing machines (NTMs), and nondeterministic oracle Turing machines (oracle NTMs) and their computational complexity. The reader is referred to standard textbooks such as Hopcroft and Ullman (1979) for the exact definitions.

We denote by POLY the set of polynomials with nonnegative integer coefficients. Let P and NP be the classes of sets (of binary strings) accepted by polynomial-time TMs and NTMs, respectively. Let A be a set and C a class of sets. We let P^A and NP^A denote the classes of sets accepted by polynomial-time oracle TMs and oracle NTMs, respectively, with oracle A. Let $\mathrm{P}^C = \bigcup \{\mathrm{P}^A\colon A \in C\}$ and $\mathrm{NP}^C = \bigcup \{\mathrm{NP}^A\colon A \in C\}$. The relativized polynomial-time hierarchy may be defined as follows (Stockmeyer, 1976):

$$\sum_0^{\mathrm{P},A} = \prod_0^{\mathrm{P},A} = \Delta_0^{\mathrm{P},A} = \mathrm{P}^A,$$

and for $k \geq 1$,

$$\sum_k^{\mathrm{P},A} = \mathrm{NP}^{\Sigma_{k-1}^{\mathrm{P},A}}, \qquad \prod_k^{\mathrm{P},A} = \mathrm{co}\text{-}\sum_k^{\mathrm{P},A} \qquad \text{and} \qquad \Delta_k^{\mathrm{P},A} = P^{\Sigma_{k-1}^{\mathrm{P},A}}.$$

When $A = \emptyset$, we have the unrelativized polynomial hierarchy, and write Σ_k^{P}, Π_k^{P}, and Δ_k^{P} for $\Sigma_k^{\mathrm{P},\emptyset}$, $\Pi_k^{\mathrm{P},\emptyset}$, and $\Delta_k^{\mathrm{P},\emptyset}$. Baker, Gill, and Solovay (1975) showed that there exist oracles A and B such that $\mathrm{P}^A = \mathrm{NP}^A$ but $\mathrm{P}^B \neq \mathrm{NP}^B \neq \mathrm{co\text{-}NP}^B$. Baker and Selman (1979) showed the existence of an oracle C such that $\Sigma_2^{\mathrm{P},C} \neq \Pi_2^{\mathrm{P},C}$.

We will use binary strings to represent real numbers. Define D to be the set of all dyadic rational numbers, i.e., all rational numbers that have finite binary expansions. A dyadic rational d is represented by a string s in the set $S = \{+, -\}\{0, 1\}^* \cdot \{0, 1\}^*$. More precisely, if $s = \pm d_n \cdots d_1 d_0 \cdot e_1 \cdots e_m$ then it represents the dyadic rational $d = \pm(\sum_{i=0}^{n} d_i \cdot 2^i + \sum_{j=1}^{m} e_j \cdot 2^{-j})$. In most cases, we do not distinguish a dyadic rational number from its representation. For each string s, we write $\|s\|$ to denote its length. (Note that for a real number x, $|x|$ denotes the absolute value of x.) For each string s in S, we write $\mathrm{prec}(s)$ to denote the number of bits to the right of the binary point of s. Sometimes we write, for d in D, $\mathrm{prec}(d) \leq n$ to mean that d has a representation s with $\mathrm{prec}(s) \leq n$ (i.e., $d = m/2^n$ for some integer m), and define D_n to be the set of all dyadic rationals d with $\mathrm{prec}(d) \leq n$.

We use two representations of real numbers: *Cauchy sequences* and *Dedekind cuts*. We say a function $\varphi: N \to D$ *binary converges* to a real number x if, for each $n \in N$, $\varphi(n)$ has precision n (i.e., $\text{prec}(\varphi(n)) \leq n$) and $|\varphi(n) - x| \leq 2^{-n}$. We let $\text{CS}(x)$, for each real x, be the collection of all functions that binary converge to x. For each real number x and each $\varphi \in \text{CS}(x)$, let $L_\varphi = \{s \in S: s \leq \varphi(\text{prec}(s))\}$. We say L_φ is a left cut of x and write $L_\varphi \in \text{LC}(x)$. We note that L_φ is polynomial-time reducible to φ: to determine whether $s \in L_\varphi$, where $\text{prec}(s) = n$, we need only find $\varphi(n)$ and compare to see whether $s \leq \varphi(n)$. Conversely, $\varphi(n) = \max\{s \in D_n: s \in L_\varphi\}$ can be found by a binary search over the set $D_n \cap L_\varphi$. If $\varphi \in \text{CS}(x)$ has the property that for all n, $\varphi(n) \leq x < \varphi(n) + 2^{-n}$, then we say φ is a *standard Cauchy sequence* for x, and the associated left cut L_φ is a *standard left cut* of x. An important property of the standard left cut L_0 of x is that $L_0 \subseteq L$ for all $L \in \text{LC}(x)$.

Now we are ready to define the computational complexity of real numbers and real functions.

DEFINITION 1. (i) A real number x is *computable* if there is a function $\varphi \in \text{CS}(x)$ that is computable.

(ii) A real number x is *polynomial-time computable* if there is a function $\varphi \in \text{CS}(x)$ that is computable in polynomial time (i.e., $\varphi(n)$ is computable in time $p(n)$ for some $p \in \text{POLY}$).

(iii) A real number x is *NP-time computable* (called an *NP real number*) if there is a set $L \in \text{LC}(x)$ that is in NP (i.e., there is a NTM M accepting L and for each $s \in L$, $M(s)$ halts in $p(\|s\|)$ moves for some $p \in \text{POLY}$).

The computational model for real functions is the oracle TM. We consider only real functions defined on the unit interval $[0,1]$.

DEFINITION 2. (i) A real function $f: [0, 1] \to R$ is *computable* if there is an oracle TM M that uses a function $\varphi: N \to D$ as a function oracle and takes an integer n as input, such that for any $x \in [0, 1]$ and any $\varphi \in \text{CS}(x)$, the function $\psi(n)$ computed by $M^\varphi(n)$ is in $\text{CS}(f(x))$.

(ii) A real function $f: [0, 1] \to R$ is *polynomial-time computable* (or, $f \in \text{P}_{C[0,1]}$) if f is computable by a polynomial-time oracle TM M (i.e., for any $x \in [0, 1]$, any $\varphi \in \text{CS}(x)$, and any integer n, $M^\varphi(n)$ halts in $p(n)$ moves for some $p \in \text{POLY}$).

(iii) A real function $f: [0, 1] \to R$ is *NP-time computable* (or, $f \in \text{NP}_{C[0,1]}$) if there is a polynomial-time oracle NTM M that uses a function $\varphi: N \to D$ as a function oracle and takes a string $d \in D$ as an input, such that for any $x \in [0, 1]$ and any $\varphi \in \text{CS}(x)$, the set $\{d \in D: M^\varphi(d)$ accepts$\} \in \text{LC}(f(x))$.

In other words, $f \in \text{P}_{C[0,1]}$ if there is a polynomial time operator mapping each function $\varphi \in \text{CS}(x)$ to a function $\psi \in \text{CS}(f(x))$ for all $x \in [0, 1]$; and $f \in \text{NP}_{C[0,1]}$ if there is a NP-time operator mapping each function $\varphi \in \text{CS}(x)$

to a set $L \in \mathrm{LC}(f(x))$ for each $x \in [0, 1]$. Since for each $\psi \in \mathrm{CS}(f(x))$, its associated left cut L_ψ is polynomial-time computable relative to ψ, it is obvious that $\mathrm{P}_{C[0,1]} \subseteq \mathrm{NP}_{C[0,1]}$.

A very useful piecewise linear function characterization of computable and polynomial-time computable real functions was first proved by Shepherdson (1976) and Ko and Friedman (1982).

PROPOSITION 1. *A real function f: $[0, 1] \to R$ is computable iff there is a sequence of piecewise linear functions $\{f_n\}$, and a recursive function m, such that*

(i) (simple piecewise linearity) *for each n, the set of breakpoints of f_n is a subset of $D_{m(n)}$;*

(ii) (uniform modulus) *for each n and each $d \in D \cap [0, 1)$ with* $\mathrm{prec}(d) \leq m(n)$, $|f_n(d) - f_n(d + 2^{-m(n)})| \leq 2^{-n}$;

(iii) (uniform convergence) *for each n and each $x \in [0, 1]$,* $|f_n(x) - f(x)| \leq 2^{-n}$;

(iv) (uniform computability) *the function g: $N \times D \to D$, defined by*

$$g(n, d) = \begin{cases} f_n(d) & \text{if } d \in D_{m(n)} \cap [0, 1], \\ 0 & \text{otherwise} \end{cases}$$

is computable.

PROPOSITION 2. *A real function f: $[0, 1] \to R$ is polynomial-time computable iff there is a sequence of piecewise linear functions $\{f_n\}$, and a function $m \in$ POLY, such that*

(i), (ii), (iii) (*as in Proposition* 1) *and*

(iv′) *the function $g(n, d)$ defined in* (iv) *above is computable in time $\leq q(n)$ for some $q \in$ POLY.*

We can extend the above characterizations to NP-time computable real functions.

PROPOSITION 3. *A real function f: $[0, 1] \to R$ is* NP*-time computable iff there is a sequence of piecewise linear functions $\{f_n\}$, and a function $m \in$ POLY, such that*

(i), (ii), (iii) (*as in Proposition* 1) *and*

(iv″) *the set $A = \{(0^n, d, e)$: $d \in D_{m(n)} \cap [0, 1]$, $e \in D_n$, and $e \leq f_n(d)\} \in$ NP. (More precisely, we mean that $A' = \{(0^n, t, s)$: $s, t \in S$,* $\mathrm{prec}(s) = n$, $\mathrm{prec}(t) = m(n)$, $t \in [0, 1]$, *and $s \leq f_n(t)\} \in$ NP.)*

Sketch of Proof. The proofs of Propositions 1, 2, and 3 are similar. We only give a short sketch of Proposition 3.

First, assume that a sequence of piecewise linear functions $\{f_n\}$ exists and satisfies (i)–(iii) and (iv″). We consider the oracle NTM M with the following algorithm:

With oracle φ and input $e \in D_n$, M queries oracle to get $d = \varphi(m(n+2))$ and accepts e iff $(0^{n+2}, d, e - 2^{-(n+1)}) \in A$.

By conditions (ii) and (iii), we know that $|f(x) - f_{n+2}(d)| \leq 2^{-(n+1)}$ if $\varphi \in \mathrm{CS}(x)$. Since M^φ accepts e iff $e - 2^{-(n+1)} \leq f_{n+2}(d)$, we have $|e' - f(x)| \leq 2^{-n}$, where $e' = \max\{e \in D_n : M^\varphi \text{ accepts } e\}$. Thus $\{e \in D: M^\varphi \text{ accepts } e\} \in \mathrm{LC}(f(x))$.

Conversely, assume that M is a polynomial-time oracle NTM such that M^φ accepts a left cut $f(x)$ whenever $\varphi \in \mathrm{CS}(x)$. Let $t \in \mathrm{POLY}$ be a time bound for M and let $m(n) = t(n+1)$. Then m satisfies the following condition:

$$|x - y| \leq 2^{-m(n)} \text{ implies } |f(x) - f(y)| \leq 2^{-(n+1)}. \qquad (*)$$

For each $d \in D_{m(n)} \cap [0, 1]$, let φ be the standard Cauchy sequence for d. Define $f_n(d) = \max\{e \in D_n : M^\varphi \text{ accepts } e\}$. Then it is clear that $\{f_n\}$ and m satisfy conditions (i)–(iii) and (iv″). ■

The condition (*) in the above proof is an important property of functions in $\mathrm{P}_{C[0,1]}$ and $\mathrm{NP}_{C[0,1]}$. It is shown in Ko and Friedman (1982) that a real function f has a polynomifally bounded modulus function m (i.e., f satisfies condition (*) with $m \in \mathrm{POLY}$) iff f is polynomial-time computable relative to an oracle set E.

Two-dimensional polynomial-time computable real functions are similarly defined. Namely, a real function f: $[0, 1]^2 \to R$ is in $\mathrm{P}_{C[0,1]^2}$ if there is a polynomial-time oracle TM M which uses two oracle functions φ and ψ, such that the function $\theta(n) = M^{\varphi,\psi}(n)$ binary converges to $f(x, y)$ if $\varphi \in \mathrm{CS}(x)$ and $\psi \in \mathrm{CS}(y)$. The proof of the following proposition is similar to the one-dimensional case and is omitted.

PROPOSITION 4. *A real function* f: $[0, 1]^2 \to R$ is in $\mathrm{P}_{C[0,1]^2}$ *iff there is a sequence of piecewise linear functions* $\{f_n\}$ *on* $[0, 1]^2$, *and a function* $m \in \mathrm{POLY}$, *such that*

(i) *for each* n, *the breakpoints of* f_n *are* $\{(d, e)$: $d, e \in D_{m(n)} \cap [0, 1]\}$,

(ii) *for each* n *and each* $d, e \in D_{m(n)} \cap [0, 1]$

$$|f_n(d, e) - f_n(d, e + 2^{-m(n)})| \leq 2^{-n} \qquad \text{if } e \neq 1.0,$$

$$|f_n(d, e) - f_n(d + 2^{-m(n)}, e)| \leq 2^{-n} \qquad \text{if } d \neq 1.0,$$

(iii) *for each* n *and* $x, y \in [0, 1]$,

$$|f_n(x, y) - f(x, y)| \leq 2^{-n},$$

and

(iv) *the function* g, *defined by*

$$g(0^n, d, e) = f_n(d, e) \qquad \text{if } d, e \in D_{m(n)} \cap [0, 1],$$
$$= 0 \qquad \text{otherwise,}$$

is polynomial-time computable.

3. Maximization on Real Functions

In this section we consider the maximization problem on real functions. Our main result is the characterization of the computational complexity of the maximization problem by the concept of NP-time computable real functions.

We first recall that for computable real functions, the maximum value $y = \max\{f(x): x \in [0, 1]\}$ of a computable real function f must be computable; on the other hand, Specker (1959) showed that there exists a computable real function f on $[0, 1]$ that does not take its maximum at any computable real number. At the polynomial-time level, the set of maximum values of polynomial-time computable real functions are not known to be polynomial-time computable. The best we can say about their complexity is that they constitute the set of all NP real numbers.

THEOREM 5. (Ko, 1982). *A real number* x *is* NP*-time computable iff there is a function* $g \in P_{C[0,1]}$ *such that* $x = \max\{g(y): y \in [0, 1]\}$.

Sketch of Proof. Since the proofs of Theorems 6 and 7 are similar to but more complicated than this proof, we reprove this result first. Our proof here uses Propositions 2 and 3 and is simpler than the original one.

First, the "if" direction is straightforward, following from the existential quantifier characterization of the class NP. We will show only the "only if" direction.

Assume that L is a left cut of x and L is in NP. (We emphasize that L is a set of *strings* in $S = \{+, -\}\{0, 1\}^* \cdot \{0, 1\}^*$ with the property that if $\text{prec}(d) = \text{prec}(e)$ and $d \leq e$ then $e \in L$ implies $d \in L$). We also assume, without loss of generality, that $\frac{1}{4} \leq x \leq \frac{3}{4}$. By the existential quantifier characterization of the class NP, there exist a polynomial-time predicate R, and a function $p \in \text{POLY}$, such that for all $d \in D \cap [0, 1]$, if $d = 0 \cdot t$ for some $t \in \{0, 1\}^*$, then

$$d \in L \qquad \text{iff } (\exists s, \|s\| = p(\|t\|))R(t, s).$$

(Note that for each string $t \in \{0, 1\}^*$, we write $0 \cdot t$ to denote the dyadic rational whose binary representation is $0 \cdot t$.) We assume that $p(n) > p(n - 1)$ for all $n \geq 1$.

We will construct a sequence of linear piecewise functions $\{g_n\}$ on $[0, 1]$ with the following properties:

(i) The set of breakpoints of g_n is $B_n = \{d \in D_{q(n)} \cap [0, 1]$: the last $2(n + p(n))$ bits of d are in $\{0\}^*$ or $\{01, 10\}^*\}$, where $q(n) = \sum_{i=1}^{n} 2(i + p(i))$.

(ii) If $d_1, d_2 \in D_{q(n)} \cap [0, 1]$ and $|d_1 - d_2| = 2^{-q(n)}$, then $|g_n(d_1) - g_n(d_2)| \leq 2^{-(n-1)}$.

(iii) If $d \in D_{q(n)} \cap [0, 1]$, then $|g_n(d) - g_{n+1}(d)| \leq 2^{-(n-1)}$.

(iv) The function $\lambda 0^n, d[g_n(d)]$ is polynomial-time computable.

Note that by Proposition 2, $\{g_n\}$ converges to a function $g \in P_{C[0,1]}$.

The idea of the construction of $\{g_n\}$ is to decode each input dyadic rational d to try to find a pair (t, s) with the property $R(t, s)$. If such a pair is found, $g_n(d)$ outputs the dyadic rational $0 \cdot t$ as long as the distance between $0 \cdot t$ and $g_{n-1}(d)$ is not too large to affect the convergence requirement (iii). We note that the choice of the set of breakpoints B_n is made to satisfy the continuity requirement (ii).

We first define a simple translation function τ that maps 0 to 01 and 1 to 10, and define $g_0(x) := 0$ for all $x \in [0, 1]$.

The definition of g_n can be described as follows:

(1) On input $d \in B_n$, we first decode d to get v, $w \in \{0, 1\}^*$ with $\|v\| = q(n - 1)$, $\|w\| = 2(n + p(n))$, and $d = 0 \cdot vw$.

(2) If $w = 0^{2(n+p(n))}$ then let $g_n(d) := g_{n-1}(0 \cdot v)$;

(3) Otherwise we further decode w into t, $s \in \{0, 1\}^*$ with $\|t\| = n$, $\|s\| = p(n)$, and $w = \tau(t)\tau(s)$, and test $R(t, s)$.

(3.1) If not $R(t, s)$, then we simply let $g_n(d) := g_{n-1}(0 \cdot v)$,

(3.2) Otherwise, if $R(t, s)$ then define

$$
\begin{aligned}
g_n(d) &= g_{n-1}(0 \cdot v) + 2^{-(n-1)} && \text{if } g_{n-1}(0 \cdot v) + 2^{-(n-1)} \leq 0 \cdot t - 2^{-n}, \\
&= 0 \cdot t - 2^{-n} && \text{if } g_{n-1}(0 \cdot v) \leq 0 \cdot t - 2^{-n} \\
& && < g_{n-1}(0 \cdot v) + 2^{-(n-1)}, \\
&= g_{n-1}(0 \cdot v) && \text{otherwise.}
\end{aligned}
$$

The above description gives a straightforward polynomial-time algorithm for computing $g_n(d)$ for $d \in B_n$, and so condition (iv) is satisfied. We check conditions (ii) and (iii) below.

Condition (ii). Let $d_1, d_2 \in B_n$, $d_1 < d_2$ and in between d_1 and d_2 there is no other point in B_n. Then, $|d_1 - d_2| \geq 2^{-q(n)}$. We consider two cases.

Case 1. The first $q(n - 1)$ bits of d_1 and d_2 agree.

That is, $d_1 = 0 \cdot vw_2$ and $d_2 = 0 \cdot vw_2$ for some v of length $q(n - 1)$ and some w_1 and w_2 of length $q(n) - q(n - 1)$. In this case, $|g_n(d_1) - g_n(d_2)| \leq 2^{-(n-1)}$ because both values are bounded by $g_{n-1}(0 \cdot v)$ and $g_{n-1}(0 \cdot v) + 2^{-(n-1)}$.

Case 2. Not Case 1.

Then, it must be the case that $d_1 = 0 \cdot v_1 w_1$ and $d_2 = 0.v_2 w_2$, where $0 \cdot v_2 = 0 \cdot v_1 + 2^{-q(n-1)}$, $w_1 = (10)^{n+p(n)}$, and $w_2 = 0^{2(n+p(n))}$. By induction, $|g_{n-1}(0 \cdot v_1) - g_{n-1}(0 \cdot v_2)| \leq 2^{-(n-2)}$. So, $|g_n(d_1) - g_n(d_2)| \leq 2^{-(n-3)}$, because $g_n(d_2) = g_{n-1}(0 \cdot v_2)$ and $|g_n(d_1) - g_{n-1}(0 \cdot v_1)| \leq 2^{-(n-1)}$. Note that the distance between d_1 and d_2, in this case, is at least $2^{-(q(n-1)+2)}$. So, for any d_3 and d_4 with distance $2^{-q(n)}$, $|g_n(d_3) - g_n(d_4)| \leq 2^{-(n-3)}2^{-2} \leq 2^{-(n-1)}$, since we assumed that $p(n) > p(n-1)$.

Condition (iii). Note that if $d = 0 \cdot vw$ with $\|v\| = q(n-1)$ and $\|w\| = q(n)$ then $g_{n-1}(0 \cdot v) \leq g_n(d) \leq g_{n-1}(0 \cdot v) + 2^{-(n-1)}$. So, by condition (ii) above, $|g_{n-1}(d) - g_n(d)| \leq 2^{-(n-2)}$.

Finally we check that $\max\{g(y) : y \in [0, 1]\} = x$. Let φ be the standard Cauchy sequence for x. Then, for each $n \geq 1$, $\varphi(n) = 0 \cdot t_n \leq x < 0 \cdot t_n + 2^{-n}$, for some $t_n \in \{0, 1\}^*$ and $0 \cdot t_n \in L$. (Recall that the standard left cut of x is contained in every left cut of x.) So, there exists, for each n, an s_n such that $\|s_n\| = p(n)$ and $R(t_n, s_n)$. Let $d_n = 0 \cdot \tau(t_1)\tau(s_1) \cdots \tau(t_n)\tau(s_n)$. Then $g_n(d_n) = 0 \cdot t_n - 2^{-n}$ because by induction $0 \cdot t_{n-1} - 2^{-(n-1)} = g_{n-1}(d_{n-1})$ and so $g_{n-1}(d_{n-1}) + 2^{-(n-1)} = 0 \cdot t_{n-1} \geq 0 \cdot t_n - 2^{-n}$. So, $g(\lim_{n\to\infty} d_n) = x$, and $\max\{g(y): y \in [0, 1]\} \geq x$.

Conversely, we can easily show, by induction, that $g_n(d) \leq 0 \cdot t_n$ for all $d \in B_n$. So, $\max\{g(y): y \in [0.1]\} \leq x$ and the proof is complete. ■

The above theorem characterizes the set of maximum values of real functions as the set of NP real numbers. However, the exact relation between the complexity of NP real numbers and the P = ?NP question is still not clear. We discuss this question in Section 5. The next theorem generalizes Theorem 5 to two-dimensional real functions and gives a characterization of NP-time computable real functions. It also generalizes Friedman's (1984) result that the maximum problem of a polynomial-time computable two-dimensional real function is always solvable in polynomial time iff P = NP.

Since we are concerned with computable, continuous functions with domain [0,], we will assume, for the sake of simplicity, that the ranges of these functions are contained in [0,]. Also, in the rest of the paper, we write D and D_n to denote $D \cap [0, 1]$ and $D_n \cap [0, 1]$, respectively.

THEOREM 6. *A real function* $f: [0, 1] \to [0, 1]$ *is* NP*-time computable iff there is a polynomial-time computable real function* $g: [0, 1]^2 \to [0, 1]$ *such that for all* $x \in [0, 1]$, $f(x) = \max\{g(x, y): y \in [0, 1]\}$.

Proof. (*If*) Assume that M is a (two-)oracle TM computing g in time $p(n)$ for some $p \in$ POLY. Then we can define a sequence of piecewise linear functions $\{f_n\}$ as follows.

For each n, the set of breakpoints of f_n is exactly $D_{p(n+2)}$. For each $d \in D_{p(n+2)}$, define $f_n(d)$ to be $\max\{M^{d,e}(n+2): e \in D_{p(n+2)}\}$, where

$M^{d,e}(n+2)$ means that the oracles φ and ψ used by M are the standard Cauchy sequences for d and e, respectively.

Now we check that $\{f_n\}$ satisfies conditions (i)–(iii) and (iv″) of Proposition 3. Condition (i) follows immediately from the definition. Condition (iv″) is also obvious by observing that for all $d \in D_{p(n+2)}$, and $e' \in D_{n+2}$, $e' \leq f_d(d)$ iff $(\exists e \in D_{p(n+2)})\ e' \leq M^{d,e}(n+2)$.

Condition (ii). If $d' = d + 2^{-p(n+2)}$ then, for all $e \in D_{p(n+2)}$, $|M^{d,e}(n+2) - M^{d',e}(n+2)| \leq 2^{-(n+1)}$ because both values are close to $g(d, e)$ within an error of $2^{-(n+2)}$. It follows from the definition of f_n that $|f_n(d) - f_n(d')| \leq 2^{-(n+1)}$.

Condition (iii). Let y_x be a maximum point for the function $\lambda y[g(x, y)]$ and define $f(x) = g(x, y_x)$. First observe that the continuity of g implies the continuity of f: if $|x - x'| \leq 2^{-p(n+2)}$ then $f(x) = g(x, y_x) \leq g(x', y_x) + 2^{-(n+2)} \leq g(x', y_{x'}) + 2^{-(n+2)} = f(x') + 2^{-(n+2)}$; and vice versa, $f(x') \leq f(x) + 2^{-(n+2)}$. Now for each $d \in D_{p(n+2)}$ we have $|f_n(d) - g(d, y_d)| \leq 2^{-(n+2)}$. Therefore, $|f_n(x) - f(x)| \leq |f_n(x) - f_n(d)| + |f_n(d) - f(d)| + |f(d) - f(x)| \leq 2^{-(n+1)} + 2^{-(n+2)} + 2^{-(n+2)} = 2^{-n}$, where d is a point in $D_{p(n+2)}$ with $|d - x| \leq 2^{-p(n+2)}$.

(*Only if*). This part of the proof is an extension of that of Theorem 5. First, from Proposition 2, there exists a sequence of piecewise linear functions $\{f_n\}$ such that conditions (i)–(iii) and (iv″) are satisfied by $\{f_n\}$. In particular, the breakpoints of f_n are in $D_{m(n)}$ for some $m \in$ POLY and the set $A = \{(0^n, d, e)\colon d \in D_{m(n)}, e \in D_n \text{ and } e \leq f_n(d)\} \in$ NP. From condition (iii), $e \leq f_n(d)$ implies $e \leq f(d) + 2^{-n}$. By the existential quantifier characterization of NP, let R be a polynomial-time predicate such that

$(0^n, d, e) \in A$

iff $e = 0 \cdot s \leq f_n(0 \cdot t) = f_n(d)$, where s and t are strings of lengths n and $m(n)$, respectively,

iff $(\exists u, \|u\| = p(n))\ R(0^n, s, t, u)$,

where p is a function in POLY and $p(n) > p(n-1)$.

Now we describe the function g_n as follows.

$$g_0(x, y) = 0 \qquad \text{for all } x, y \in [0, 1].$$

The set of breakpoints of g_n, $n \geq 1$, is $D_{m(n)} \times B_n$, where $B_n = \{d \in D_{q(n)}$: the last $q(n) - q(n-1)$ bits of d are in $\{0\}^*$ or $\{01, 10\}^*\}$ and $q(n) = \sum_{i=1}^{n} 2(i + p(i) + m(i))$. On input $(d_1, d_2) \in D_{m(n)} \times B_n$, $g_n(d_1, d_2)$ is defined as follows.

(1) First decode d_2 as $0 \cdot vw$ with $\|v\| = q(n-1)$ and $\|w\| = q(n) - q(n-1)$.

(2) If $w = 0^{q(n)-q(n-1)}$ then let $g_n(d_1, d_2) := g_{n-1}(d_1, 0 \cdot v)$.

(3) Otherwise we decode $w = \tau(t)\tau(u)\tau(s)$ with $\|s\| = n$, $\|t\| = m(n)$, and $\|u\| = p(n)$ and test $R(0^n, s, t, u)$.

(3.1) If not $R(0^n, s, t, u)$ then let $g_n(d_1, d_2) := g_{n-1}(d_1, 0 \cdot v)$.

(3.2) Otherwise, compute $e' := 0 \cdot s - |d_1 - 0 \cdot t| 2^{m(n)-n} - 2^{-(n-1)}$, and let

$$
\begin{aligned}
g_n(d_1, d_2) &= g_{n-1}(d_1, 0 \cdot v) + 2^{-(n-4)} && \text{if } g_{n-1}(d_1, 0 \cdot v) + 2^{-(n-4)} \le e', \\
&= e' && \text{if } g_{n-1}(d_1, 0 \cdot v) \le e' \\
& && < g_{n-1}(d_1, 0 \cdot v) + 2^{-(n-4)}, \\
&= g_{n-1}(d_1, 0 \cdot v) && \text{otherwise.}
\end{aligned}
$$

We note that the definition of g_n is similar to that of g_n in the proof of Theorem 5, and we need to verify that $\{g_n\}$ satisfies the conditions (i)–(iv) of Proposition 4. We note that conditions (i) and (iii) and the first inequality of (ii) can be proved similarly to that of the proof of Theorem 5, because for each fixed x, the function $\lambda y[g_n(x, y)]$ is essentially the same as g_n defined in Theorem 5.

Condition (ii). We need to verify the second inequality of (ii). In other words, if $d_1, d_2 \in D_{m(n)}$, $d_3 \in D_{q(n)}$, and $d_2 = d_1 + 2^{-m(n)}$, then $|g_n(d_1, d_3) - g_n(d_2, d_3)| \le 2^{-(n-3)}$. The main idea here is that in step (3.2) of the definition of g_n, when we decode d_3 and find witness u for the relation $[0 \cdot s \le f_n(0 \cdot t)]$, if $0 \cdot t = d_1$ then we make $g_n(d_1, d_3) = 0 \cdot s - 2^{-(n-1)}$ to make max g_n at d_1 close to $f_n(d_1)$, and even if $0 \cdot t = d_2 \neq d_1$, we still make the value of $g_n(d_1, d_3)$ as close to $0 \cdot s - 2^{-(n-1)}$ as possible to satisfy the continuity requirement for g.

More precisely, we assume, without loss of generality, that $m(n) > m(n-1) + 1$, and also assume, by induction, that $|g_{n-1}(d_1, d_3) - g_{n-1}(d_2, d_3)| \le 2^{-(n-4)}2^{-2} = 2^{-(n-2)}$. (Note that $|d_1 - d_2| \le 2^{-2}2^{-m(n-1)}$.) Now, decode $d_3 = 0 \cdot vw$ with $\|v\| = q(n-1)$ and $\|w\| = q(n) - q(n-1)$.

If $[w = 0^{q(n)-q(n-1)}]$ or $[w = \tau(t)\tau(u)\tau(s)$ and not $R(0^n, s, t, u)]$ then $g_n(d_1, d_3) = g_{n-1}(d_1, 0 \cdot v)$ and $g_n(d_2, d_3) = g_{n-1}(d_2, 0 \cdot v)$, and so $|g_n(d_1, d_3) - g_n(d_2, d_3)| \le 2^{-(n-2)}$.

Otherwise (i.e., $R(0^n, s, t, u)$), then $|\,|0 \cdot t - d_1| - |0 \cdot t - d_2|\,| = |d_1 - d_2| = 2^{-m(n)}$. So, $|e_1' - e_2'| = 2^{-n}$, where $e_i' = 0 \cdot s - |0 \cdot t - d_i| \cdot 2^{m(n)-n} - 2^{-(n-1)}$, for $i = 1, 2$. It follows from the definition and the inductive hypothesis that $|g_n(d_1, d_3) - g_n(d_2, d_3)| \le |g_{n-1}(d_1, 0 \cdot v) - g_{n-1}(d_2, 0 \cdot v)| + |e_1' - e_2'| \le 2^{-(n-2)} + 2^{-n} \le 2^{-(n-3)}$.

Condition (iv). Following the definition of g_n, we have the following algorithm for computing $g_n(d_1, d_2)$ with $d_1 \in D_{m(n)}$ and $d_2 \in D_{q(n)}$:

First let

$$\text{lower}(d_2) := \max\{d \in B_n;\ d \le d_2\},$$

$\text{upper}(d_2) := \min\{d \in B_n\colon d \geq d_2\}$,
$\text{left}(d_1) := \max\{d \in D_{m(n-1)}\colon d \leq d_1\}$,
$\text{right}(d_1) := \min\{d \in D_{m(n-1)}\colon d \geq d_1\}$,
$\text{leftlow}(d_2) := \max\{d \in D_{q(n-1)}\colon d \leq \text{lower}(d_2)\}$,
$\text{leftup}(d_2) := \max\{d \in D_{q(n-1)}\colon d \leq \text{upper}(d_2)\}$,
and let r_1 and r_2 be dyadic rationals in $[0, 1]$ such that
$d_1 = (1 - r_1) \cdot \text{left}(d_1) + r_1 \cdot \text{right}(d_1)$ and
$d_2 = (1 - r_2) \cdot \text{lower}(d_2) + r_2 \cdot \text{upper}(d_2)$.
(Note that it is possible that $\text{left}(d_1) = \text{right}(d_1)$, and in this case we let $r_1 := 0$. The same rule applies for r_2.)

Next define
$e_1 := g_{n-1}(\text{left}(d_1), \text{leftlow}(d_2))$,
$e_2 := g_{n-1}(\text{left}(d_1), \text{leftup}(d_2))$,
$e_3 := g_{n-1}(\text{right}(d_1), \text{leftlow}(d_2))$,
$e_4 := g_{n-1}(\text{right}(d_1), \text{leftup}(d_2))$.
Then we have
$e_5 := g_{n-1}(d_1, \text{leftlow}(d_2)) = (1 - r_1) \cdot e_1 + r_1 \cdot e_3$, and
$e_6 := g_{n-1}(d_1, \text{leftup}(d_2)) = (1 - r_1) \cdot e_2 + r_1 \cdot e_4$.
Following the definition of g_n, we can compute, using e_5 and e_6,
$e_7 := g_n(d_1, \text{lower}(d_2))$ and
$e_8 := g_n(d_1, \text{upper}(d_2))$.
Finally, we get
$g_n(d_1, d_2) := (1 - r_2) \cdot e_7 + r_2 \cdot e_8$.

We note that in the above algorithm we made four recursive calls to the algorithm itself for the values e_1, e_2, e_3, and e_4. (Note that $\text{left}(d_1)$ and $\text{right}(d_1)$ are in $D_{m(n-1)}$ and $\text{leftlow}(d_2)$ and $\text{leftup}(d_2)$ are in $D_{q(n-1)}$.) Therefore, if the algorithm is called recursively, then we would have to make 4^{n-1} many calls to reach the trivial case of evaluating g_0, and this would give an exponential-time algorithm. To avoid this problem, we can use the concept of dynamic programming to actually call $g_{n-2}(d_1, d_2)$, with $d_1 \in D_{m(n-2)}$ and $d_2 \in D_{q(n-2)}$, only four times. To see this, we observe that for any $d_1 \in D_{m(n)}$, $\text{left}(\text{left}(d_1)) = \text{left}(\text{right}(d_1))$, $\text{right}(\text{left}(d_1)) = \text{right}(\text{right}(d_1))$, and for any $d_2 \in D_{q(n)}$, $\text{leftlow}(\text{leftlow}(d_2)) = \text{leftlow}(\text{leftup}(d_2))$ and $\text{leftup}(\text{leftlow}(d_2)) = \text{leftup}(\text{leftup}(d_2))$. Therefore we can use a bottom-up algorithm to compute $g_i(\text{left}_i(d_1), \text{leftlow}_i(d_2))$, $g_i(\text{left}_i(d_1), \text{leftup}_i(d_2))$, $g_i(\text{right}_i(d_1), \text{leftlow}_i(d_2))$ and $g_i(\text{right}_i(d_1), \text{leftup}_i(d_2))$ for $i := 0$ to n, where $\text{left}_i(d_1)$ is defined to be $\max\{d \in D_{m(i)}\colon d \leq d_1\}$, and $\text{right}_i(d_1)$, $\text{leftlow}_i(d_2)$ and $\text{leftup}_i(d_2)$ are similarly defined. This proves the polynomial-time computability of the function $\lambda 0^n, d_1, d_2[g_n(d_1, d_2)]$.

Finally we need to check that for each x, $f(x) = \max\{g(x, y)\colon y \in [0, 1]\}$. For each $x \in [0, 1]$, let x_n be the maximum dyadic rational in $D_{m(n)}$ that is $\leq x$, and $x_n' = x_n + 2^{-m(n)}$. We note that for each n, $|f_n(x_n) - f(x)| \leq$

$2^{-(n-1)}$. We claim that for each $n \in N$ and $d \in B_n$, $g_n(x_n, d) \leq f(x)$ and $g_n(x_n', d) \leq f(x)$.

We prove this claim by induction on n.

First, if $g_n(x_n, d) = g_{n-1}(x_n, 0 \cdot v)$, where v is the first $q(n-1)$ bits of d, then, by the inductive hypothesis and the piecewise linearity of g_{n-1}, $g_n(x_n, d) \leq \max\{g_{n-1}(x_{n-1}, d)g_{n-1}(x_{n-1}', d)\} \leq f(x)$.

Otherwise, if $g_n(x_n, d) > g_{n-1}(x_n, 0 \cdot v)$, then when we decode d into $0 \cdot v\tau(t)\tau(u)\tau(s)$ we have $R(0^n, s, t, u)$ and $g_n(x_n, d) \leq e' = 0 \cdot s - |x_n - 0 \cdot t| \cdot 2^{m(n)-n} - 2^{-(n-1)}$. Since f_n has a modulus of continuity $m(n)$ and since f_n is piecewise linear, we have $|f_n(x_n) - f_n(0 \cdot t)| \leq 2^{-n} \cdot |x_n - 0 \cdot t| \cdot 2^{m(n)}$. So, $g_n(x_n, d) \leq 0 \cdot s - |f_n(x_n) - f_n(0 \cdot t)| - 2^{-(n-1)} \leq f_n(0 \cdot t) - |f_n(x_n) - f_n(0 \cdot t)| - 2^{-(n-1)} \leq f_n(x_n) - 2^{-(n-1)} \leq f(x)$. $g_n(x_n', d) \leq f(x)$ can be proved similarly.

Therefore, we have, for every $x, y \in [0, 1]$, $g(x, y) = \lim_{n\to\infty} g_n(x_n, y) \leq f(x)$.

Next, consider $d_n = 0 \cdot \tau(x_1)\tau(t_1)\tau(f_1(x_1)) \cdots \tau(x_n)\tau(t_n)\tau(f_n(x_n))$, and claim that $g_n(x_n, d_n) \geq f_n(x_n) - 2^{-(n-1)}$.

Assume, by induction, that $g_{n-1}(x_{n-1}, d_{n-1}) \geq f_{n-1}(x_{n-1}) - 2^{-(n-2)}$. Then, $g_{n-1}(x_n, d_{n-1}) \geq f_{n-1}(x_{n-1}) - 2^{-(n-3)} - 2^{-(n-2)}$ (by the second part of condition (ii) of $\{g_n\}$) and $\geq f_n(x_n) - 2^{-(n-1)} - 2^{-(n-4)}$ (by the continuity and convergence of $\{f_n\}$). Furthermore, the value e' evaluated in step (3.2) of the definition of g_n is equal to $f_n(x_n) - 2^{-(n-1)}$. So, from the definition of g_n (step (3.2)), $g_n(x_n, d_n) \geq f_n(x_n) - 2^{-(n-1)}$. From the claim, we have $g(x, \lim_{n\to\infty} d_n) = \lim_{n\to\infty} g_n(x_n, d_n) \geq x$. This completes the proof of Theorem 6. ■

4. A Polynomial Hierarchy of Real Functions

In Section 3, we gave a characterization of NP-time computable real functions in terms of the maximization operation. We note that the proof technique of this characterization relativizes; that is, we can actually prove that, for any set E, f is NP-time computable on $[0, 1]$ relative to E iff there is a function g that is polynomial time computable on $[0, 1]^2$ relative to E, such that $f(x) = \max\{g(x, y): y \in [0, 1]\}$. This suggests that we may extend the class of NP-time computable real functions to a polynomial hierarchy and provide a natural characterization for this hierarchy by the alternating operations of maximization and minimization. We discuss this idea in this section.

First we define the polynomial hierarchy of real functions. It is natural to define $\Sigma^{\mathrm{P}}_{k,C[0,1]}$, $k \geq 1$, to be the class of real functions f on $[0, 1]$ that is computable by a polynomial-time two-oracle NTM M with respect to an oracle $B \in \Sigma^{\mathrm{P}}_{k-1}$, in the sense that M uses B and a function $\varphi \in \mathrm{CS}(x)$ as oracles and the set $\{d \in D: M^{\varphi,B} \text{ accepts } d\} \in \mathrm{LC}(f(x))$. It is straightforward to relativize Proposition 3 to get the following equivalent definition. We use the following definition because it is more useful in proving the characterization results.

DEFINITION 3. For $k \geq 0$, we say a real function f: $[0, 1] \to [0, 1]$ is in $\Sigma^{\mathrm{P}}_{k,C[0,1]}$ ($\Pi^{\mathrm{P}}_{k,C[0,1]}$, $\Delta^{\mathrm{P}}_{k,C[0,1]}$) if there exists a sequence of piecewise linear functions $\{f_n\}$, and a function $m \in$ POLY, such that

(i), (ii), and (iii) of Proposition 1 hold, and

(iv) the set $A = \{(0^n, d, e)$: $d \in D_{m(n)}$, $e \in D_n$, and $e \leq f_n(d)\} \in \Sigma^{\mathrm{P}}_k(\Pi^{\mathrm{P}}_k, \Delta^{\mathrm{P}}_k$, respectively).

Using this definition, we generalize Theorem 6 to the following:

THEOREM 7. *Let $k \geq 0$. A real function f: $[0, 1] \to [0, 1]$ is in $\Sigma^{\mathrm{P}}_{k+1,C[0,1]}$ iff there is a real function g: $[0, 1]^2 \to [0, 1]$ that is in $\Pi^{\mathrm{P}}_{k,C[0,1]^2}$ such that for all $x \in [0, 1]$, $f(x) = \max\{g(x, y)$: $y \in [0, 1]\}$.*

Proof. The proof is almost the same as that of Theorem 6.

First, $f \in \Sigma^{\mathrm{P}}_{k+1,C[0,1]}$ implies that $0 \cdot s \leq f_n(0 \cdot t)$ iff $(\exists u, \|u\| = p(n))$ $R(0^n, s, t, u)$, for some Π^{P}_k-predicate R. Now define $\{g_n\}$ exactly the same as that in the proof of Theorem 6. Then, it follows from the same proof that $\{g_n\}$ satisfies (i)–(iii) of Proposition 4 and $\{g_n\}$ converges to some function g such that for all x, $f(x) = \max\{f(x, y)$: $y \in [0, 1]\}$. The only thing left to show is that the set $B = \{(0^n, e, d_1, d_2)$: $e \in D_n$, $d_1 \in D_{m(n)}$, $d_2 \in D_{q(n)}$, and $e \leq g_n(d_1, d_2)\} \in \Pi^{\mathrm{P}}_k$. (The proof of Theorem 6 gave $B \in \Delta^{P}_{k+1}$.) We will use the notation defined in the proof of Theorem 6.

From the definition of g_n, to compute $g_n(d_1, d_3)$, with $d_1 \in D_{m(n)}$ and $d_3 \in B_n$, it involves the decoding of d_3 to extract the strings s, t, u, and v and the dyadic rational e'. This decoding process takes only a polynomially bounded amount of time. Using these values we can simplify the predicate $[e \leq g_n(d_1, d_3)]$ with $e \in D_n$, $d_1 \in D_{m(n)}$, and $d_3 \in B_n$, as follows: (Note that $\mathrm{leftlow}(d_3) = \mathrm{leftup}(d_3) = 0 \cdot v$, because $d_3 \in B_n$.)

$e \leq g_n(d_1, d_3)$
iff $[e \leq g_{n-1}(d_1, \mathrm{leftlow}(d_3))]$ or $[R(0^n, s, t, u)$ and $e \leq e'$ and $e - 2^{-(n-4)} \leq g_{n-1}(d_1, \mathrm{leftlow}(d_3))]$,

because $g_n(d_1, d_3) > g_{n-1}(d_1, \mathrm{leftlow}(d_3))$ implies $g_n(d_1, d_3) = \min\{e', g_{n-1}(d_1, \mathrm{leftlow}(d_3)) + 2^{-(n-4)}\}$.

The predicate $[e \leq g_{n-1}(d_1, d_2')]$, with $d_1 \in D_{m(n)}$ and $d_2' \in D_{q(n-1)}$ can be further simplified as follows:

$e \leq g_{n-1}(d_1, d_2')$
iff $\forall e_1 \forall e_2[\ [e_1 > g_{n-1}(\mathrm{left}(d_1), d_2')$ and $e_2 > g_{n-1}(\mathrm{right}(d_1), d_2')] \Rightarrow$
$e \leq (1 - r_1)e_1 + r_1 e_2]$
iff $\forall e_1 \forall e_2[e_1 \leq g_{n-1}(\mathrm{left}(d_1), d_2')$ or $e_2 \leq g_{n-1}(\mathrm{right}(d_1), d_2')$ or $e \leq (1 - r_1)e_1 + r_1 e_2]$.

Together, we have

$e \leq g_n(d_1, d_3)$
iff $\forall e_1 \forall e_2[e_1 \leq g_{n-1}(\mathrm{left}(d_1), \mathrm{leftlow}(d_3))$ or $e_2 \leq g_{n-1}(\mathrm{right}(d_1), \mathrm{leftlow}(d_3))$

or $e \leq (1 - r_1)e_1 + r_1e_2$ or $[R(0^n, s, t, u)$ and $e \leq e'$ and $e - 2^{-(n-4)} \leq (1 - r_1)e_1 + r_1e_2]$]. (1)

Now consider $[e \leq g_n(d_1, d_2)]$, with $e \in D_n$, $d_1 \in D_{m(n)}$ and $d_2 \in D_{q(n)}$. We have

$e \leq g_n(d_1, d_2)$
iff $\forall e_7 \forall e_8 [e_7 \leq g_n(d_1,$ lower$(d_2))$ or $e_8 \leq g_n(d_1,$ upper$(d_2))$ or $e \leq (1 - r_2)e_7 + r_2e_8]$. (2)

Combining (1) and (2), we can reduce $[e \leq g_n(d_1, d_2)]$ to a predicate of the following form:

$e \leq g_n(d_1, d_2)$
iff $\forall e_1 \forall e_2 \forall e_3 \forall e_4 [e_1 \leq g_{n-1}($left$(d_1),$ leftlow$(d_2))$ or $e_2 \leq g_{n-1}($right$(d_1),$ leftlow$(d_2))$ or $e_3 \leq g_{n-1}($left(d_1) leftup$(d_2))$ or $e_4 \leq g_{n-1}($right$(d_1),$ leftup$(d_2))$ or $S(e_1, e_2, e_3, e_4, e)]$ (3)

where $S(e_1, e_2, e_3, e_4, e)$ is a predicate of the form

$$\forall e_7 \forall e_8 [A_1 \text{ or } A_2 \text{ or } A_3 \text{ or } [R_1 \text{ and } A_4] \text{ or } [R_2 \text{ and } A_5]\ ]$$

and A_i's are polynomial-time predicates and R_j's are Π_k^P-predicates, $i = 1, \ldots, 5$ and $j = 1, 2$.

When we simplify (3) recursively, we need, at each step, only to introduce four more universal quantifiers and four more S-type predicates, because of the relations such as left(left(d_1)) = left(right(d_1)), as discussed in the proof of condition (iv) in Theorem 6.

$e \leq g_n(d_1, d_2)$
iff $\forall e_1 \cdots \forall e_4 \forall e_1' \cdots \forall e_4' [Q_1$ or $\cdots$ or Q_4 or S_1 or $\cdots$ or S_4 or $S]$,

where Q_i is of the type $e_i' \leq g_{n-2}(d_1'', d_2'')$ with $d_1'' \in D_{m(n-2)}$ and $d_2'' \in D_{q(n-2)}$, $i = 1, 2, 3, 4$; and $S_i = S(e_1', e_2', e_3', e_4', e_i)$, $i = 1, 2, 3, 4$; and $S = S(e_1, e_2, e_3, e_4, e)$.

A complete expansion will then produce a predicate of $4n$ universal quantifiers followed by $4n - 3$ S-type predicates. Since each S-type predicate contains two universal quantifiers and two occurrences of R-type predicates, it shows that the predicate $[e \leq g_n(d_1, d_2)]$ is in Π_k^P, and this completes the proof. ■

The proofs of Theorems 6 and 7 can also be generalized to multi-dimensional functions. In other words, a real function f: $[0, 1]^m \to [0, 1]$, for some $m \geq 1$, is in $\Sigma_{k+1, C[0,1]^m}^P$ iff $f(x_1, \ldots, x_m) = \max\{g(x_1, \ldots, x_m, y)$: $y \in [0, 1]\}$ for some $g \in \Pi_{k, C[0,1]^{m+1}}^P$. (The only difference in proof from that of Theorem 7 is that when the function g_n is constructed, we need to make it to be continuous in all $m + 1$ variables. This is easy to handle using the technique used in step (3.2) of the definition of g_n in Theorem 6.) Thus, we have the following corollary.

COROLLARY 8. *Let $k \geq 1$. A real function f: $[0, 1] \to [0, 1]$ is in $\Sigma^{\mathrm{P}}_{k, C[0,1]}$ iff there is a function $g \in \mathrm{P}_{C[0,1]^{k+1}}$ such that for all $x \in [0, 1]$,*

$$f(x) = \max_{y_1 \in [0,1]} \min_{y_2 \in [0,1]} \cdots \operatorname{opt}_{y_k \in [0,1]} g(x, y_1, \ldots, y_k),$$

where $\operatorname{opt}_k = \max$ *if k is odd* ($= \min$ *if k is even*).

5. STRUCTURE OF THE POLYNOMIAL HIERARCHY OF REAL FUNCTIONS

In Section 4, we defined a polynomial hierarchy of real functions and showed a natural correspondence between this hierarchy and the operations of maximization and minimization. In this section we study the structure of this hierarchy. In particular, we ask whether the hierarchy is an infinite hierarchy. The answer is that for every $k \geq 1$, $\Sigma^P_k = \Pi^P_k$ iff $\Sigma^{\mathrm{P}}_{k, C[0,1]} = \Pi^{\mathrm{P}}_{k, C[0,1]}$; and $P = \mathrm{NP}$ iff $P_{C[0,1]} = \mathrm{NP}_{C[0,1]}$.

First we note that the following theorem follows immediately from Definition 3.

THEOREM 9. *For each $k \geq 1$, $\Sigma^{\mathrm{P}}_k = \Pi^{\mathrm{P}}_k$ implies $\Sigma^{\mathrm{P}}_{k, C[0,1]} = \Pi^{\mathrm{P}}_{k, C[0,1]}$*

We recall that Baker, Gill, and Solovay (1975) and Baker and Selman (1979) have shown that the polynomial hierarchy of operators extends at least two levels. Thus, Theorem 9 shows that the polynomial hierarchy of real functions is different from the polynomial hierarchy of operators. We note that the difference lies on the inherent structure of real numbers. In the following we illustrate, in general, how the structure of left cuts affects the oracle computation when they are used as oracles. In particular, we show that a left cut, when used as an oracle, cannot distinguish NP-operators from P-operators, unless $\mathrm{P} \neq \mathrm{NP}$. To begin with, we review some definitions.

DEFINITION 4. (Karp and Lipton, 1980). A set $A \subseteq \{0, 1\}^*$ has *polynomial-size circuits*, or $A \in \mathrm{P/poly}$, if there are a polynomial length-bounded function h: $\{0\}^* \to \{0, 1\}^*$ and a set $B \in \mathrm{P}$, such that for each $s \in \{0, 1\}^*$,

$$s \in A \text{ iff } \langle s, h(0^{\|s\|}) \rangle \in B.$$

Meyer has shown that $A \in \mathrm{P/poly}$ iff there is a sparse set S such that $A \in \mathrm{P}^S$ (Berman and Hartmanis, 1977). (A set $S \subseteq \{0, 1\}^*$ is *sparse* if there is a function $p \in \mathrm{POLY}$ such that for every n, the size of the set $\{s \in S: \|s\| = n\}$ is bounded by $p(n)$.) Karp and Lipton (1980) and Sipser showed that a set A in P/poly cannot be NP-complete unless $\Sigma^{\mathrm{P}}_2 = \Pi^{\mathrm{P}}_2$. Ko (1983) showed that a left cut of a real number is in P/poly. In the following, we give a stronger result.

DEFINITION 5. A set $A \subseteq \{0, 1\}^*$ has *self-producible* (*polynomial-size*) *circuits* if $A \in \mathrm{P/poly}$ and the function h in Definition 4 is in P^A.

In other words, A has self-producible circuits if the circuits $h(0^n)$ that help the computation of "$s \in A$" for those s with $\|s\| = n$ can be computed from A itself. It is easy to show that a left cut L of a real number has self-producible circuits.

LEMMA 10. (a) *If A is a tally language (i.e., $A \subseteq \{0\}^*$), then A has self-producible circuits.*

(b) *If L is a left cut of a real number, then L has self-producible circuits.*

Proof. (a) Define $h(0^n) = 1$ if $0^n \in A$, and $= 0$ if $0^n \notin A$, and let $B = \{\langle 0^n, 1\rangle : n \geq 1\}$ (cf. Long and Selman, 1983).

(b) Define $h(0^n) = \max\{d \in D_n : d \in L\}$, and $B = \{\langle e, d\rangle : \text{prec}(e) = \text{prec}(d) \text{ and } e \leq d\}$. Note that h can be computed from L by a binary search, because a left cut L has the property that

$$[\text{prec}(d) = \text{prec}(e),\ e \leq d \text{ and } d \in L] \text{ implies } e \in L. \quad \blacksquare$$

Now, our result follows from Long and Selman's (1983) observation that a set having self-producible circuits cannot distinguish NP-operators from P-operators.

THEOREM 11. (Long and Selman, 1983). *For all $k \geq 1$, $\Sigma_k^P = \Pi_{kP}^P$ implies $\Sigma_k^{P,A} = \Pi_k^{P,A}$ for all A that have self-producible circuits.*

Sketch of Proof. The main observation is that $B \in \Sigma_k^{P,A}$ iff there is a polynomial length-bounded function h: $\{0\}^* \to \{0, 1\}^*$, and a set $C \in \Sigma_k^P$, such that for all $s \in \{0, 1\}^*$, $s \in B$ iff $\langle s, h(0^{\|s\|})\rangle \in C$. Furthermore, if A has self-producible circuits then $h \in P^A$. This is because the oracle A can be replaced by circuits h without increasing the complexity of B.

Now, $\Sigma_k^P = \Pi_k^P$ implies that the set C in the above is also in Π_k^P, and hence $B \in \Pi_k^{P,A}$, because $h \in P^A$. $\blacksquare$

COROLLARY 12. *Let L be a left cut of a real number. Then, for $k \geq 1$, $\Sigma_k^P = \Pi_k^P$ implies $\Sigma_k^{P,L} = \Pi_k^{P,L}$; and $P = NP$ implies $P^L = NP^L$.*

Conversely, we want to show that $\Sigma_{k,C[0,1]}^P = \Pi_{k,C[0,1]}^P$ implies $\Sigma_k^P = \Pi_k^P$. We first examine the simplest case.

THEOREM 13. $P_{C[0,1]} = NP_{C[0,1]}$ *implies* $P = NP$.

Sketch of Proof. Friedman (1984) showed that for every set $A \in NP$, there is a real function $g_A \in P_{C[0,1]^2}$ such that the function $f(x) = \max\{g(x, y) : y \in [0, 1]\}$ is in $P_{C[0,1]}$ iff $A \in P$. So, it follows from Theorem 6 that $P_{C[0,1]} = NP_{C[0,1]}$ implies $P = NP$. We can also give a direct proof that is similar to but simpler than the proof of Theorem 6.

Let $A \in NP$. We construct a function $f \in NP_{C[0,1]}$ that embeds the set A such that $f \in P_{C[0,1]}$ iff $A \in P$. Since $A \in NP$, there exists a polynomial-time predicate R, and a function $p \in POLY$, such that $s \in A$ iff $(\exists t, \|t\| = p(\|s\|))R(s, t)$. We construct a sequence of piecewise linear functions $\{f_n\}$ as follows.

The breakpoints of f_n are dyadic rationals in $B_n = \{d \in D_{2n}: d = 0 \cdot vw$ for some $v \in \{01, 10\}^*$ and $w \in \{0\}^*\}$. Note that $d \in B_n - B_{n-1}$ means that $d = 0 \cdot v$ for some $v \in \{01, 10\}^*$.

We let $f_0(x) = x$ for all $x \in [0, 1]$. For $n \geq 1$, we define f_n on $d \in B_n$ as follows:

(1) For all $d \in B_{n-1}$, $f_n(d) = f_{n-1}(d)$,

(2) For all $d \in B_n - B_{n-1}$, we decode d into $\tau(t)$ (where τ is the translation function such that $\tau(0) = 01$ and $\tau(1) = 10$) and let

$$\begin{aligned} f_n(d) &= d + 2^{-(2n+1)} && \text{if } t \in A, \\ &= d && \text{otherwise.} \end{aligned}$$

It is a routine check to show that $\{f_n\}$ satisfies conditions (ii) and (iii) of Proposition 3. The key points are that

(1) for any $d \in B_k - B_{k-1}$, where $k \leq n$, we must have $d \leq f_n(d) = f_k(d) \leq d + 2^{-(2k+1)}$; and

(2) if d and e are in B_n and there is no other point in B_n between d and e and if d or e is in B_k for some $k \leq n$, then $|d - e| \geq 2^{-(2k+2)}$.

In addition, we can verify that $\{f_n\}$ satisfies condition (iv″) of Proposition 3. Let $e \in D_{2n+1}$ and $d \in B_n - B_{n-1}$. Then,

$e \leq f_n(d)$
iff $[e \leq d]$ or $[e \leq d + 2^{-(2n+1)}$ and $t \in A]$.

Now, for $e \in D_{2n+1}$ and $d \in D_{2n}$, we may use the piecewise linearity of $\{f_n\}$ to reduce this predicate as we did in the proof of Theorem 7.

$e \leq f_n(d)$
iff $(\exists e_1 \in D_{2n+1})$ $(\exists e_2 \in D_{2n+1})$ $[e_1 \leq f_n(d_1)$ and $e_2 \leq f_n(d_2)$ and $e \leq (1 - r) \cdot e_1 + r \cdot e_2]$,

where d_1, $d_2 \in B_n$ and $r \in D \cap [0, 1]$ are polynomial-time computable from d. Thus, by a reduction similar to that in the proof of Theorem 7, we can show that the predicate $[e \leq f_n(d)]$ is in NP.

Finally, it is clear that if $f \in \mathrm{P}_{C[0,1]}$, then, for each $t \in \{0, 1\}^*$ with $\|t\| = n$, we can compute $f_n(0 \cdot \tau(t))$ correct to within an error of $2^{-(2n+2)}$ and determine whether $t \in A$ according to whether $f_n(0 \cdot \tau(t)) > 0 \cdot \tau(t)$ or not. (Note that since both $f_n(0 \cdot \tau(t))$ and $0 \cdot \tau(t)$ have alternating bits 0 and 1 in every two bits, we can determine whether they are equal or not by obtaining only an approximation value of $f_n(0 \cdot \tau(t))$.) This completes the proof. ■

In the above proof, let us replace $A \in$ NP by any set A' in Σ_k^{P}, for some $k \geq 1$, and construct the function f according to A'. Then we have $f \in \Sigma_{k,C[0,1]}^{\mathrm{P}}$. Now assume that $f \in \Pi_{k,C[0,1]}^{\mathrm{P}}$. Then the predicate $[e \leq f_n(d)]$ is in Π_k^{P}, and for each t of length n, we have

$t \in A$
iff $f_n(0 \cdot \tau(t)) = 0 \cdot \tau(t) + 2^{-(2n+1)}$
iff $0 \cdot \tau(t) + 2^{-(2n+1)} \leq f_n(0 \cdot \tau(t))$.

So, $f \in \Pi^{\mathrm{P}}_{k,C[0,1]}$ implies $A \in \Pi^{\mathrm{P}}_k$. This gives the following corollary.

COROLLARY 14. *For each* $k \geq 1$, $\Sigma^{\mathrm{P}}_{k,C[0,1]} = \Pi^{\mathrm{P}}_{k,C[0,1]}$ implies $\Sigma^{\mathrm{P}}_k = \Pi^{\mathrm{P}}_k$.

Theorem 9 and Corollary 14 showed that the polynomial hierarchy of real functions, similar to the Meyer–Stockmeyer polynomial hierarchy, may collapse to any level, while the polynomial hierarchy of operators is known to have at least two proper levels. This difference is caused by the convergence requirement of the Cauchy sequence representation of real numbers. Some other effects of this structural constraint of real numbers have been discussed in Ko (1982, 1983):

(a) A left cut L of a real number cannot be NP-m-complete unless P = NP; and it cannot be NP-T-complete unless $\Sigma^{\mathrm{P}}_2 = \Pi^{\mathrm{P}}_2$.

(b) It is known that P = NP implies $\mathrm{P}_R = \mathrm{NP}_R$, and $\mathrm{P}_R = \mathrm{NP}_R$ implies EXP = NEXP, where P_R (NP_R) is the class of real numbers computable in deterministic (nondeterministic, respectively) polynomial time, and EXP (NEXP) is the class of sets (of binary strings) acceptable in deterministic (nondeterministic, respectively) time $O(2^{cn})$ for some constant c. It is not known, however, whether $\mathrm{P}_R = \mathrm{NP}_R$ implies P = NP. In fact, this question is equivalent to one of the major open questions in complexity theory.

The reader is referred to these papers for details.

6. CONCLUSION

We have demonstrated the close connection between maximization and nondeterminism, in both discrete complexity theory and continuous complexity theory. A polynomial hierarchy of real functions is constructed as a continuous analog of the Meyer–Stockmeyer polynomial hierarchy. It was shown that the complexity of the alternating operations of maximization and minimization has a natural characterization by this hierarchy.

It should be pointed out, however, that our formulation of the problem is of an abstract form, while most practical problems are of more complex forms. Nevertheless, our results may give a general direction and, in some cases, may simplify the complexity analysis of the maximization problems. One example is that the computational complexity of the degrees of best approximation by polynomials can be characterized exactly to be NP real numbers. This result will be discussed in a subsequent paper.

In Section 5, we touched briefly upon the structure of real numbers and its effect on the complexity of real functions. This issue appears important both in the complexity theory of real computation and in the structural study of

discrete polynomial complexity theory. Ko (1984) discussed further the role a left cut construct plays in the structural studies of polynomial-time reducibilities and of relativization. This direction of research provides another interesting aspect of the interconnections between discrete complexity theory and continuous complexity theory.

REFERENCES

BAKER, T., GILL, J., AND SOLOVAY, R. (1975), Relativizations of the P=?NP question, *SIAM J. Comput.* **4**, 431–442.

BAKER, T., AND SELMAN, A. (1979), A second step toward the polynomial hierarchy, *Theoret. Comput. Sci.* **8**, 177–187.

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

FRIEDMAN, H. (1984), On the computational complexity of maximization and integration, *Adv. in Math.* **53**, 80–98.

GAREY, M., AND JOHNSON, D. (1979), "Computers and Intractability," Freeman, San Francisco.

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

JOHNSON, D. (1974), Approximation algorithms for combinatorial problems, *J. Comput. System Sci.* **9**, 256–278.

KARP, R., AND LIPTON, R. (1980), Some connections between nonuniform and uniform complexity classes, in "Proceedings, 12th ACM Symposium on Theory of Computing," pp. 302–309.

KO, K. (1982), The maximum value problem and NP real numbers, *J. Comput. System Sci.* **24**, 15–35.

KO, K. (1983), On self-reducibility and weak p-selectivity, *J. Comput. System Sci.* **26**, 209–221.

KO, K. (1984), Applying techniques of discrete complexity theory to numerical computation, *in* "Studies in Complexity Theory" (R. Book, Ed.), Pitman, London, in press.

KO, K. (1985), On some natural complete operators, *Theoret. Comput. Sci.* **37**, 1–30.

KO, K. AND FRIEDMAN, H. (1982), Computational complexity of real functions, *Theoret. Comput. Sci.* **20**, 323–352.

LONG, T., AND SELMAN, A. (1983), Relativizing complexity classes with sparse oracles, *J. Assoc. Comput. Mach.*, in press.

SHEPHERDSON, J. (1976), On the definition of computable functions of a real variable, *Z Math. Logik Grundlag. Math.* **22**, 391–402.

SPECKER, E. (1959), Der Satz vom Maximum in der rekursiven Analysis, *in* "Constructivity in Mathematics" (A. Heyting, Ed.), pp. 254–265, North-Holland, Amsterdam.

STOCKMEYER, L. (1976), The polynomial-time hierarchy, *Theoret. Comput. Sci.* **3**, 1–22.