JOURNAL OF COMPLEXITY **11**, 265–292 (1995)

Computational Complexity of Fixed Points and Intersection Points

KER-I KO*

Department of Computer Science, State University of New York at Stony Brook, Stony Brook, New York 11794

Received December 14, 1994

We study the computational complexity of Brouwer's fixed point theorem and the intersection point theorem in the two-dimensional case. Papadimitriou (1990, *in* "Proceedings, 31st IEEE Sympos. Found. Comput. Sci.," pp. 794–801) defined a complexity class *PDLF* to characterize the complexity of the fixed point theorem in the three-dimensional case. We define a subclass *PMLF* of *PDLF* and show that the fixed points and the intersection points of polynomial-time computable functions are not polynomial-time computable if *PMLF* contains a function on unary inputs that is not polynomial-time computable. © 1995 Academic Press, Inc.

1. INTRODUCTION

Brouwer's *fixed point theorem* states that a continuous function f mapping the d-dimensional unit cube $[0, 1]^d$ to itself must have a fixed point $\mathbf{z}_0 \in [0, 1]^d$ such that $f(\mathbf{z}_0) = \mathbf{z}_0$ (see, e.g., Shashkin, 1991). A variation of the fixed point theorem for the case of dimension $d = 1$ is the following *intersection point theorem:* Let Γ_1 and Γ_2 be two continuous, simple curves lying inside the unit square $[0, 1]^2$ such that Γ_1 connects the left side of the unit square to the right side and Γ_2 connects the upper side of the unit square to the lower side. Then, Γ_1 and Γ_2 must have an intersection point. Brouwer's fixed point theorem for dimension $d = 1$ is a special case of the intersection point theorem with Γ_1 being the graph of a continuous function f mapping $[0, 1]$ to itself and Γ_2 being the graph of the function $g(x) = x$. In this paper, we investigate the computational complexity of fixed points and intersection points when the underlying functions and curves are polynomial-time computable.

* Research supported in part by NSF Grant CCR 9121472. E-mail: keriko@sbcs.sunysb.edu.

The complexity of finding the fixed points has been studied in two different computational models for real functions. In the *continuous real RAM model,* the exact arithmetic and comparison operations on real numbers can be performed in one unit of time, and the function f is presented as an oracle that answers any query for $f(\mathbf{z})$ with its exact real value. Hirsch *et al.* (1989) proved, in the real RAM model, an exponential lower bound $2^{O(n+d)}$ for finding an *approximate fixed point* $\mathbf{z}$ such that $|f(\mathbf{z}) - \mathbf{z}| \leq 2^{-n}$, for continuous functions f that map $[0, 1]^d$ to itself. In the *discrete Turing machine model,* the function f is presented by a polynomial-time Turing machine (TM) M that, on any dyadic point $\mathbf{z}$ of precision n, gives an approximate value $M(\mathbf{z})$ for $f(\mathbf{z})$ with $|M(\mathbf{z}) - f(\mathbf{z})| \leq 2^{-n}$. Both the machine M and the error bound 2^{-n} are given as the input to the problem. Papadimitriou (1990) showed in this model that the problem of finding an approximate fixed point for the cases of dimension $d \geq 3$ is complete for the complexity class *PDLF.* The class *PDLF* is the class of search problems for each of its instances there exists a (possibly exponentially long) locally polynomial-time computable search path leading an initial point to the solution point. Thus, this class captures the inherent complexity of the combinatorial search of Sperner's lemma. The case of dimension $d = 2$ in the discrete Turing machine model was left open.

In this paper, we study these problems in the *continuous Turing machine model* of Ko and Friedman (1982) (see also Ko, 1991). That is, we use ordinary Turing machines that work with finite strings as the computational model, but we require that the output values of a Turing machine for computing a real number converge in a polynomial speed. In this model, a real number x is called *polynomial-time computable* if there exists a Turing machine M that, on input n, outputs a dyadic rational d in time polynomial in n such that $|d - x| \leq 2^{-n}$. A function $f\colon [0, 1] \to \mathbf{R}$ is called *polynomial-time computable* if there is a Turing machine M that computes f as in the discrete model and f has a polynomial modulus p such that if $|x - y| \leq 2^{-p(n)}$ then $|f(x) - f(y)| \leq 2^{-n}$.

The Brouwer fixed point problem in this model is formulated as follows: if f is polynomial-time computable mapping $[0, 1]^d$ to itself, does it always have a fixed point $\mathbf{z}_0$ that is polynomial-time computable? We observe that by a simple design, it is easy to construct a polynomial-time computable function f such that it has a unique fixed point of arbitrarily high complexity (but having many approximate fixed points of low complexity). Thus, the answer to the Brouwer fixed point problem is a trivial *no,* which, however, does not provide much insight into the real combinatorial complexity of searching for the fixed points. To remedy this situation, we add an additional requirement to the function f in our study: the function $g(\mathbf{z}) = f(\mathbf{z}) - \mathbf{z}$ must have a *polynomial inverse modulus* at its zeros; that is, there exists a polynomial q such that if $|\mathbf{z} - \mathbf{z}_0| > 2^{-n}$ for all zeros $\mathbf{z}_0$ of g, then $|g(\mathbf{z})| >$

$2^{-q(n)}$ (see Ko, 1991, for more discussions on this notion and its relation to the zero-finding problem). We note that functions f satisfying this condition have the property that an approximate fixed point of the function f is indeed a good approximation to an exact fixed point of f; i.e., if $|f(\mathbf{z}) - \mathbf{z}| \le 2^{-q(n)}$ then there is a fixed point $\mathbf{z}_0$ of f such that $|\mathbf{z} - \mathbf{z}_0| \le 2^{-n}$. Thus, the complexity of the fixed points of such functions f has an upper bound NP, and the lower bound is comparable with those found in the above two different models.[1] In particular, it is easy to see that Papadimitriou's construction can be carried over to our computational model. Namely, assuming that $PDLF \not\subseteq PF$ (i.e., $PDLF$ contains a function not computable in polynomial time), then there exists a polynomial-time computable function f mapping $[0, 1]^3$ to itself such that $g(\mathbf{z}) = f(\mathbf{z}) - \mathbf{z}$ has a polynomial inverse modulus of continuity at zeros and all fixed points of f are not polynomial-time computable.

The first main result of this paper is a new lower bound for the fixed point problem in the case of dimension $d = 2$. This result is an extension of Papadimitriou's result. We define a subclass $PMLF$ of $PDLF$ to be the class of search problems whose search path is unique and is monotone under a natural ordering. We show that if $PMLF_1 \not\subseteq PF_1$ (i.e., $PMLF$ contains a function defined on unary inputs that is not computable in polynomial time), then the fixed point problem for the case $d = 2$ is not solvable in polynomial time. More precisely, we construct, under the assumption $PMLF_1 \not\subseteq PF_1$, a polynomial-time computable function f mapping $[0, 1]^2$ to itself such that $g(\mathbf{z}) = f(\mathbf{z}) - \mathbf{z}$ has a polynomial inverse modulus of continuity at zeros and f has a unique fixed point $\mathbf{z}_0$ that is not polynomial-time computable. Interestingly, the construction of the function f is based on the staircase construction of Hirsch *et al.* (1989) that established the lower bound for the fixed point theorem in the continuous RAM model.

The intersection point problem in our model can be formulated in a similar way. We say that two functions f and g mapping $[0, 1]$ to $[0, 1]^2$ are *polynomially distinguishable* away from the intersection points if for any $t_1, t_2 \in [0, 1]$, $|f(t_1) - \mathbf{z}_0| \ge 2^{-n}$ and $|g(t_2) - \mathbf{z}_0| \ge 2^{-n}$ for all intersection points $\mathbf{z}_0$ of f and g imply that $|f(t_1) - g(x)| \ge 2^{-p(n)}$ and $|f(x) - g(t_2)| \ge 2^{-p(n)}$ for some fixed polynomial p and for all $x \in [0, 1]$. Then, for such functions, an approximate intersection point is a good approximation to an exact intersection point, and the exact intersection points for functions f and g satisfying this condition can be computed in nondeterministic polynomial time. What is the exact complexity of finding an intersection point of such functions f and g? We observe that if $f(t) = \langle t, t \rangle$, then the above question is almost the same as the fixed point problem, and the binary

[1] Note that it does not make sense to consider approximate fixed points in our model, since for any function f there always exist rational approximate fixed points.

search algorithm yields a polynomial-time intersection point. Indeed, if either curve f or g is the graph of a polynomial-time computable function, then the intersection points are easy to find. However, in the general case, as functions f and g can wind around each other, the binary search algorithm apparently fails. Our second main result establishes $PMLF$ as a lower bound for the complexity of the intersection points. That is, if $PMLF_1 \not\subseteq PF_1$, then there exist functions f and g which are polynomially distinguishable away from intersection points such that f and g have a unique intersection point $\mathbf{z}_0$ that is not polynomial-time computable. This result shows a difference in complexity between the geometric-oriented method of binary search and the purely combinatorial method of Sperner's lemma for the proof of the existence of fixed points.

The above two lower bound results are based on the assumption that the class $PMLF_1$ does not collapse to the class PF_1. How plausible is this assumption? Since $PMLF$ is a subclass of $PDLF$, as well as the class UPF,[2] it is very close to PF and does not seem to have many natural complete problems. As a partial justification, we show, in the next section, that under a natural relativization, the class $PMLF$ does not collapse to the class PF relative to some oracle. It is interesting to point out that, although some of recent results such as Shamir (1990) indicated that the relativized separation does not have much to say about the unrelativized case, it does have some interesting implication on the complexity of real-valued problems. Namely, this relativized separation does imply the exponential lower bound for the fixed points in the real RAM model (this is demonstrated by the similarity between the adversary argument of our Theorem 2.4 and the proof of Hirsch *et al.*, 1989).

2. Complexity Class *PMLF*

Consider the problem of computing a multi-valued, total function $\phi: \{0, 1\}^* \rightarrow \{0, 1\}^*$. We say the function ϕ is *polynomial-time computable* (denoted by $\phi \in PF$) if there exists a *deterministic* polynomial-time TM M such that for each input x, $M(x)$ is a value of $\phi(x)$. We say ϕ is *nondeterministic polynomial-time computable* (denoted by $\phi \in TNPF$) if there exists a *nondeterministic* polynomial-time TM N such that for each input x, there exists at least one halting computation of $N(x)$ and that every halting computation path of $N(x)$ outputs a value of $\phi(x)$ (not necessarily the same one). For certain search problems, the function ϕ is not only in $TNPF$, but also has some structural properties that may help the machine to find the

[2] UPF denotes the class of single-valued total functions computable by nondeterministic polynomial-time Turing machines that have, on each input, exactly one halting computation.

solutions. For instance, Johnson *et al.* (1988) studied the class *PLS* of *polynomial-time local search problems:* A local search problem is a multi-valued optimization function ϕ for which there exists a deterministic polynomial-time TM M that on input (x, y) either verifies that y is an optimal solution to $\phi(x)$ or outputs a new candidate z as a *better* solution to $\phi(x)$ than y. Note that for each local search problem ϕ, there is a *generic* search path for each input x: starting with any potential solution y_0, then at each iteration applying machine M to (x, y_i) to either, verifies that y_i is an optimal solution, or to get a better solution y_{i+1}. This algorithm, although not necessarily halting in polynomial time, demonstrates a structural property of the problem ϕ and, hence, suggests that problem ϕ is probably not complete for the more general complexity class *TNPF*. Johnson *et al.* (1988) and Papadimitriou (1990) argued that the problems with the same structural properties ought to be grouped together as a new complexity class, and they proved that such classes possess many natural complete problems.

In the following, we formally define complexity classes *PDLF* and *PMLF* of search problems that have a property similar to that of polynomial-time local search problems. Let n be a positive integer and $V = \{0, 1\}^n$. We say a directed graph $G = (V, E)$ is a *path graph*, if for any node $w \in V$, both the outdegree of w and the indegree of w are bounded by 1. Thus, each node w has at most one predecessor and one successor. A path graph G is called *monotone* if $(u, v) \in E$ implies that $u < v$ under the lexicographic ordering on $\{0, 1\}^n$. Let $A = V \cup \{\lambda\}$, where λ denotes the empty string. We say that a function $\psi: V \to A \times A$ *represents* the path graph G if for any $w \in V$, $\psi(w) = (u, v)$, where u is the (unique) predecessor of w and v is the successor of w, and u (and v) is λ if w does not have a predecessor (successor, respectively). The class *PDLF*, as defined in Papadimitriou (1990), is the class of all multi-valued functions $\phi: \{0, 1\}^* \to \{0, 1\}^*$ for which there exist a polynomial-time computable function $\psi: \{0, 1\}^* \times \{0, 1\}^* \to \{0, 1\}^* \times \{0, 1\}^*$ and a polynomial p such that for each string x of length n,

(i) the function $\psi_x(y) = \psi(x, y)$, when restricted to inputs y of length $p(n)$, represents a path graph G_x on $\{0, 1\}^{p(n)}$ of which $0^{p(n)}$ is a leaf, and

(ii) $\phi(x) = \{w \in \{0, 1\}^{p(n)}: w \text{ is a leaf of } G_x, w \neq 0^{p(n)}\}$.

We define a new complexity class *PMLF* (*M* stands for *monotone*) as the class of *single-valued* functions $\phi: \{0, 1\}^* \to \{0, 1\}^*$ in *PDLF* having the following additional properties:

(iii) for each string x of length n, $\psi_x(y)$ represents a monotone path graph G_x on $\{0, 1\}^{p(n)}$;

(iv) each G_x has exactly two leaves: $0^{p(n)}$ and $\phi(x)$.

In other words, the value $\phi(x)$ in *PMLF* can be computed by following

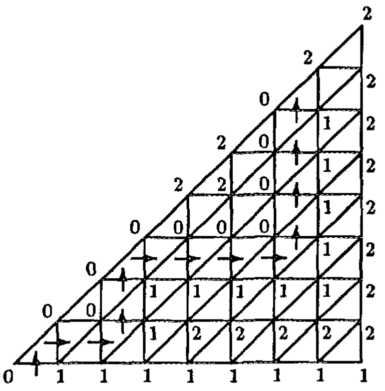

FIG. 1. Triangle T_3 with a coloring θ_x, with respect to the sequence $y_0 = 0$, $y_1 = 2$ and $y_2 = y_{k_x} = 6$.

the path $y_0 < y_1 < \cdots < y_k$, where $y_0 = 0^{p(n)}$ and y_{i+1} is the successor of y_i in the path graph represented by ψ_x, and $\phi(x)$ is the point y_k that does not have a successor. This search path is in general, however, exponentially long, and so it does not provide a polynomial-time search algorithm. It is, of course, computable in polynomial time by a nondeterministic algorithm that guesses the point y_k and verifies that y_k does not have a predecessor. Indeed, this witness is unique by our definition and so it is actually contained in the class *UPF*. This observation proves the following relation between the classes *PMLF* and *PF*. In the following, *UP* denotes the complexity class of sets that are acceptable by *unambiguous* nondeterministic TMs in polynomial time that on any input have at most one halting computation path (Ko, 1991).

PROPOSITION 2.1. *If $PMLF \not\subseteq PF$ then $P \neq UP \cap co\text{-}UP$.*

Proof. Assume that $\phi \in PMLF - PF$. Let $A = \{\langle x, u\rangle : u$ is a prefix of $\phi(x)\}$. Then it is clear that $A \in UP \cap co\text{-}UP$, since for each x we can guess a unique string w and verify in polynomial time that $w = \phi(x)$ and that u is a prefix (or, not a prefix) of w. Since we can use A as an oracle to compute ϕ in polynomial time, the set A is not in P. ∎

The class *PMLF* is defined to capture the inherent complexity of the fixed point theorem in the two-dimensional plane. Let us illustrate this idea by showing that *PMLF* is a lower bound for the two-dimensional Sperner lemma as defined in Papadimitriou (1990). Our result on the two-dimensional fixed point problem is based on this construction.

We consider a simplified form of Sperner's lemma on the triangle T_m with the following three vertices: $\langle 0, 0\rangle$, $\langle 2^m, 0\rangle$, and $\langle 2^m, 2^m\rangle$. The triangle T_m is divided into 2^{2m} subtriangles as shown in Fig. 1. Each subtriangle either has three vertices $\langle s, t\rangle$, $\langle s+1, t\rangle$, $\langle s+1, t+1\rangle$, with $0 \le t \le s \le 2^m - 1$, or it has three vertices $\langle s, t-1\rangle$, $\langle s, t\rangle$, $\langle s+1, t\rangle$, with $1 \le t \le s \le 2^m - 1$. We say that $\theta:\{0, 1, \ldots, 2^m\} \times \{0, 1, \ldots, 2^m\} \to \{0, 1, 2\}$ defines an *admissible coloring* on triangle T_m if

(i) $\theta(0, 0) = 0$, $\theta(2^m, 0) = 1$, $\theta(2^m, 2^m) = 2$,

(ii) $\theta(s, 0) \in \{0, 1\}$, $\theta(s, s) \in \{0, 2\}$, $\theta(2^m, t) \in \{1, 2\}$, for all $s, t \in \{0, 1, \ldots, 2^m\}$.

The problem SPERNER$_2$ is as follows: Given a polynomial-time TM M and an integer m, such that M on inputs $(s, t) \in \{0, 1, \ldots, 2^m\} \times \{0, 1, \ldots, 2^m\}$ defines an admissible coloring on T_m, find a subtriangle of T_m whose vertices are colored by three different colors. (The existence of such a subtriangle is guaranteed by the Sperner lemma.) Papadimitriou (1990) showed that SPERNER$_d$ is in class *PDLF* for all $d \ge 2$ and that SPERNER$_d$ is complete for *PDLF* for all $d \ge 3$. In the following, we show that SPERNER$_2$ is not polynomial-time solvable if $PMLF \not\subseteq PF$.

THEOREM 2.2. *If $PMLF \not\subseteq PF$, then* SPERNER$_2$ *is not polynomial-time computable.*

Proof. Assume that $\phi \in PMLF - PF$. Thus, for each x of length n, there is an increasing sequence of strings $y_0 = 0^{p(n)} < y_1 < \cdots < y_{k_x}$ of length $p(n)$ such that $\phi(x) = y_{k_x}$ and the mapping from (x, y_i) to (y_{i-1}, y_{i+1}) is computable in polynomial time. In the following, we identify each string y_i with the integer j, $0 \le j \le 2^{p(n)} - 1$, whose $p(n)$-bit binary representation is y_i. Note that for infinitely many x, the value y_{k_x} is less than $2^{p(n)} - 1$, since y_{k_x} would be easy to compute otherwise. For each x with $y_{k_x} \le 2^{p(n)} - 2$, we define a coloring θ_x on triangle $T_{p(n)}$ as follows:

(1) $\theta_x(0, 0) = 0$, $\theta_x(s, 0) = 1$ if $1 \le s \le 2^{p(n)}$, and $\theta_x(2^m, t) = 2$ if $1 \le t \le 2^{p(n)}$.

(2) $\theta_x(s, t) = 1$ if $t = y_i$ and $y_i + 1 \le s \le y_{i+1} + 1$ for some $0 \le i \le k_x - 1$, or if $s = y_i + 1$ and $y_{i-1} \le t \le y_i$ for some $1 \le i \le k_x$.

(3) $\theta_x(s, t) = 0$ if $t = y_i + 1$ and $y_i + 1 \le s \le y_{i+1}$ for some $0 \le i \le k_x - 1$, or if $s = y_i$ and $y_{i-1} + 1 \le t \le y_i$ for some $1 \le i \le k_x$.

(4) For all other vertices (s, t) of $T_{p(n)}$, let $\theta_x(s, t) = 2$.

Figure 1 shows such a coloring.

Since the mapping from (x, y_i) to (y_{i-1}, y_{i+1}) is polynomial-time computable, it is clear that θ_x is uniformly computable in polynomial time. Furthermore, the triangle $T_{p(n)}$ with the coloring θ_x has a unique search path starting from the edge connecting $(0, 0)$ and $(1, 0)$ and continues with the edges whose two vertices are colored by 0 and 1 (shown by arrows in Fig. 1). The subtriangles in the search path, except the last triangle, are colored by colors 0 and 1. The subtriangles to the right of the search path are colored by colors 1 and 2. The subtriangles to the left of the search path are colored by colors 0 and 2. Thus, there is a unique 3-colored subtriangle in $T_{p(n)}$, namely, the one with the vertices $\langle y_{k_x}, y_{k_x}\rangle, \langle y_{k_x} + 1, y_{k_x}\rangle$, and $\langle y_{k_x} + 1, y_{k_x} + 1\rangle$. Since the vertices of this subtriangle encode the value $\phi(x) = y_{k_x}$, the problem of computing ϕ is reduced to the problem of finding the unique 3-colored subsquare, and so the theorem is proven. ∎

In Sections 4 and 5, we will use a stronger assumption than $PMLF \not\subseteq PF$ for the lower bound results. For any complexity class C of sets, we let C_1 denote the subclass of sets $A \subseteq \{0\}^*$ that is in C. For any complexity class F of functions, we let F_1 denote the subclass of functions $\phi:\{0\}^* \to \{0, 1\}^*$ that is in F. We are interested in the complexity class $PMLF_1$ of functions defined on a singleton alphabet that is in $PMLF$. The complexity of functions in $PMLF_1$ is weaker than the complexity of sets in $UP_1 \cap co\text{-}UP_1$. It is known that the complexity class $UP_1 \cap co\text{-}UP_1$ characterizes the complexity of one-way functions whose range is $\{0\}^*$ (Ko, 1991). The inverse functions ϕ^{-1} of functions ϕ in $PMLF_1$ are candidates of such one-way functions.

PROPOSITION 2.3. *If $PMLF_1 \not\subseteq PF_1$ then $P_1 \neq UP_1 \cap co\text{-}UP_1$.*

Since we do not know of any natural complete problems for the class $PMLF_1$ and yet we will use the assumption that $PMLF_1 \not\subseteq PF_1$, we need some justification for this assumption. In the following, we show that $PMLF_1$ does not collapse to PF_1 relative to some oracle. In other words, any *uniform* way of searching the second leaf of a monotone path graph must take superpolynomial time in the worst case.

First, we recall that relative to an oracle set A, the class PF^A is just all the functions computable in polynomial time by deterministic oracle TMs that use A as the oracle. The class $PMLF_1^A$ is naturally defined to be the class of functions $\phi:\{0\}^* \to \{0, 1\}^*$ for which there exist a function $\psi \in PF^A$ and a polynomial p satisfying conditions (i)–(iv) of the definition of $PMLF$.

THEOREM 2.4. *There exists a set A such that $PMLF_1^A \not\subseteq PF_1^A$.*

Proof. Let A be a subset of $\{0, 1\}^*$ and y a string of length n. We define the *A-predecessor* of y to be the least string w of length n such that $0^n1y0u \in A$ for all prefixes u of w, and the *A-successor* of y to be the last string z of length n such that $0^n1y1v \in A$ for all prefixes v of z; we say that y does not have an A-predecessor (or, an A-successor) if such a string w (or, respectively, z) does not exist. We are going to construct a set A satisfying the following conditions:

(a) For any strings y and z, y is the A-predecessor of z if and only if z is the A-successor of y.

(b) For any strings y and z, if z is the A-successor of y then $y < z$ under the lexicographic ordering on $\{0, 1\}^*$.

(c) For any string y, $0^n1y0u \in A$ if and only if u is a prefix of the A-predecessor w of y; and $0^n1y1v \in A$ if and only if v is a prefix of the A-successor z of y. (If y does not have an A-predecessor then $0^n1y0u \notin A$ for all strings u; and if y does not have an A-successor then $0^n1y1v \notin A$ for all strings v.)

(d) For each integer n, either no string of length n has an A-predecessor or an A-successor, or there is a unique A-successor chain of strings of length n, beginning with 0^n. That is, if we define a directed graph $G = (V, E)$, with $V = \{0, 1\}^n$ and $E = \{(y, z) : z \text{ is the } A\text{-successor of } y\}$, then either $E = \emptyset$ or G is a monotone path graph with exactly two leaves, one of them being 0^n.

(e) The function $\phi_A(0^n)$ = the last string in the A-successor chain of length n is not in PF^A.

The above conditions (a), (b), and (c) together imply that the A-predecessor and A-successor functions are well defined and are in PF^A. Condition (d) implies that the function ϕ_A is well defined and is in $PMLF_1^A$. Thus, the theorem is proven once set A is constructed to satisfy the above conditions.

Let $\{M_e\}$ be an effective enumeration of all polynomial-time oracle Turing machines with the runtime of M_e bounded by the eth polynomial $p_e(n) = n^e + e$. We are going to construct set A in stages. At each stage e, we construct a finite subset of A to ensure that M_e^A does not compute ϕ_A.

First, for stage 0, we let $A_0 = \emptyset$ and let $n_0 = 1$. Assume that by the end of stage $e - 1$, we have defined integer n_{e-1} and set A_{e-1}.

Stage e. We define n_e to be the least integer n such that $n > p_{e-1}(n_{e-1})$ and $p_e(n) < 2^{n-1}$. We write n for n_e. (The first condition on n guarantees that adding strings of length $\geq n$ to set A does not affect the earlier simulation of $M_{e-1}^A(0^{n_{e-1}})$. The second condition allows enough space to deny $M_e^A(0^n)$ any chance of computing $\phi_A(0^n)$.)

In the following, we describe an algorithm to construct an A-successor chain $C_n : x_{n,0} = 0^n < x_{n,1} < \cdots < x_{n,m}$ of strings of length n. Assume that a string z is greater than the last element y of the current chain C_n. We say that we *add* z to the chain C_n to mean that we add 0^n1y1u and 0^n1z0v to set A_e for all prefixes u of z and all prefixes v of y. We say that the machine M_e *queries about* the A-predecessor (or, the A-successor) of a string y if M_e asks whether $0^n1y0u \in A$ for some u of length $\leq n$ (or, respectively, asks whether $0^n1y1v \in A$ for some v of length $\leq n$).

ALGORITHM FOR THE CHAIN C_n.

(1) Let C_n consist of a single string 0^n. Let $B_n = \emptyset$.

(2) For i from 1 to $p_e(n)$ do the following:

Simulate the ith move of the computation of $M_e^A(0^n)$.

Case 1. If $M_e^A(0^n)$ does not make a query, then do nothing.

Case 2. If $M_e^A(0^n)$ queries A about a string w that is not of the form 0^n1u for any string u of length $n + 1 \leq |u| \leq 2n + 1$, then answer the query according to the set A_{e-1}.

Case 3. If $M_e^A(0^n)$ queries about the A-predecessor or the A-successor of a string y of length n that is not in C_n, then answer *no* and add y to the set B_n.

Case 4. If $M_e^A(0^n)$ queries about the A-predecessor of a string y in C_n, then answer according to the chain C_n.

Case 5. If $M_e^A(0^n)$ queries about the A-successor of a string y that is in C_n but not the last element of C_n, then answer according to the chain C_n.

Case 6. If $M_e^A(0^n)$ queries about the A-successor of the last string y in C_n, then let $z = \min\{w : |w| = n, w > y, w \notin B_n\}$, add z as the new last element of C_n, and answer the query according to the new C_n.

Case 7. If $M_e^A(0^n)$ halts and outputs a string z that is not in C_n, then add z to B_n.

(3) Let y be the last element of C_n. Define $z = \min\{w : |w| = n, w > y, w \notin B_n\}$, and add z as the new last element of C_n.

(4) Let A_e be the set A_{e-1} plus the new strings added to A in the stage e.

END OF ALGORITHM.

Note that $M_e^A(0^n)$ must halt in $p_e(n)$ moves, and so set B_n is always of size $\leq p_e(n) < 2^{n-1}$. Also, the chain C_n is also of size $\leq 2^{n-1}$. Thus, Case 6 of Step (2) and Step (3) are well defined.

We let $A = \bigcup_{e=0}^{\infty} A_e$. It is clear that through the construction of the chain C_n, conditions (a)–(d) are easily satisfied. To see that ϕ_A is not computable in polynomial time relative to A, we first verify that the simulation of $M_e^A(0^n)$ in Stage e is identical to the computation of $M_e(0^n)$ with respect to the final set A. This is true because in Stage e, we have included all queries answered with "no A-predecessor nor A-successor" in the set B_n and made sure that $B_n \cap C_n = \emptyset$. Furthermore, we never add any string of length $\leq p_e(n)$ to set A in later stages. Thus, the simulation of $M_e^A(0^n)$ in Stage e is correct and its output is different from $\phi_A(0^n)$, and condition (e) is satisfied. ∎

3. Computational Model of Real Functions

The concept of polynomial-time computable real functions used in this paper was first introduced in Ko and Friedman (1982). This concept is an extension of the notion of recursive real functions used in recursive analysis (Pour-El and Richards, 1989), based on the bit-operation complexity measure defined on Turing machines.

The basic computational objects in this model are dyadic rationals $\mathbf{D} = \{m/2^n : m \in \mathbf{Z}, n \in \mathbf{N}\}$. For each integer $n > 0$, let $\mathbf{D}_n$ denote the class of dyadic rationals with at most n bits in the fractional part of its binary representation. A real number $x \in \mathbf{R}$ is *polynomial-time computable* if there exist a Turing machine M and a polynomial p such that on the input $n \in \mathbf{N}$, the machine M outputs, in time $p(n)$, a dyadic rational d satisfying $|d - x| \le 2^{-n}$.

The notion of polynomial-time computable real functions is formally defined by oracle Turing machines. In this paper, we use an equivalent definition that avoids the use of the oracle Turing machines. We say a function $f: [0, 1] \to \mathbf{R}$ has a *polynomial modulus of continuity* if there exists a polynomial p such that $|x - y| \le 2^{-p(n)}$ implies $|f(x) - f(y)| \le 2^{-n}$.

DEFINITION 3.1. A function $f: [0, 1] \to \mathbf{R}$ is *polynomial-time computable* if

(i) f has a polynomial modulus, and

(ii) there exist a Turing machine M and a polynomial p such that for any dyadic rational number $d \in \mathbf{D}_n$, and any integer n, $M(d, n)$ outputs, in time $p(n)$, a dyadic rational number e such that $|e - f(d)| \le 2^{-n}$.

For functions f satisfying the above two conditions, it is easy to see how we can approximate the value $f(x)$ for a "given" input x: first, get a dyadic rational d that is very close to x; then use M to find an approximate value e to $f(d)$. Since d is close to x, condition (i) guarantees that e is also close to $f(x)$.

The above definition can be extended to functions mapping $[0, 1]$ to $\mathbf{R}^2$ or functions mapping $[0, 1]^2 \to \mathbf{R}^2$ in a natural way. A point in $\mathbf{R}^2$ is denoted by a bold-faced character $\mathbf{z}$ or a pair $\langle x, y\rangle$ of real numbers.

4. Complexity of Fixed Points

Let f be a polynomial-time computable function from $[0, 1]$ to itself. Then we can easily find a fixed point of f by a simple binary search method. However, in our computational model, the binary search method does not necessarily work in polynomial time, because each iteration of the binary search method involves the comparison of two real numbers, and the amount of time to perform the comparison operation could be arbitrarily high if two numbers are not identical but are arbitrarily close to each other. Or, equivalently, a point x could be very far from the exact fixed point x_0 but the function value $f(x)$ is very close to x (see Theorem 4.4 of Ko, 1991). In order to understand the inherent complexity of the combinatorial search process for fixed points, we consider a function f an *ill-conditioned* function if it has approximate fixed points that are not a good approximation to the exact fixed points. Formally, we use the notion of *polynomial inverse moduli* to identify the class of *well-conditioned* functions.

DEFINITION 4.1 (Ko, 1991). A function $f\colon [0, 1]^2 \to \mathbf{R}^2$ has a *polynomial inverse modulus at zeros* if there exist a polynomial p and a constant n_0 such that for all $n \geq n_0$, if $|\mathbf{z} - \mathbf{z}_0| > 2^{-n}$ for all zeros $\mathbf{z}_0$ of f, then $|f(\mathbf{z})| > 2^{-p(n)}$. The function p is called an *inverse modulus function of f at zeros.*

We will consider functions $f\colon [0, 1]^2 \to [0, 1]^2$ such that $g(\mathbf{z}) = f(\mathbf{z}) - \mathbf{z}$ has a polynomial inverse muludus at zeros. We call such functions f *well-conditioned* functions (with respect to the Brouwer fixed point problem). For such functions, their fixed points have an upper bound $PDLF$.

PROPOSITION 4.2. *Assume that f is a well-conditioned, polynomial-time computable function mapping* $[0, 1]^2$ *to itself. Further assume that f has a unique fixed point* $\mathbf{z}_0$. *If* $PDLF = PF$ *then* $\mathbf{z}_0$ *is polynomial-time computable.*

Proof. The proof is essentially the same as Papadimitriou's (1990) proof for the discrete Turing machine model. ■

We now show the lower bound $PMLF_1$ for the fixed points of well-conditioned, polynomial-time computable functions.

THEOREM 4.3. *Assume that* $PMLF_1 \not\subseteq PF_1$. *Then, there exists a well-conditioned, polynomial-time computable function* $f\colon [0, 1]^2 \to [0, 1]^2$ *that has a unique fixed point* $\mathbf{z}_0$ *but* $\mathbf{z}_0$ *is not polynomial-time computable.*

Proof. Assume that ϕ is a function in $PMLF_1 - PF_1$. We are going to construct a function f on $[0, 1]^2$ such that it has a unique fixed point $\mathbf{z}_0$ with the property that the binary expansion of $\mathbf{z}_0$ encodes the values $\phi(0^n)$. The construction consists of two steps. First, we describe, for each integer n, a basic construction of a function f_n on $[0, 1]^2$ such that f_n has a unique fixed point that is enclosed in a small subsquare $[a_n, b_n]^2$ with the real number a_n encoding the value $\phi(0^n)$. In the second step, basic functions f_n are combined to form the desired function f in the following way: f on $[0, 1]^2$ is identical to f_0, except on the subsquare $[a_0, b_0]^2$. Then, we embed f_1 on $[0, 1]^2$ into the subsquare $[a_0, b_0]^2$. Let $[a_1', b_1']^2$ be the image of the square

$[a_1, b_1]^2$ under this embedding function. Then, we embed f_2 on $[0, 1]^2$ into the subsquare $[a_1', b_1']^2$, etc. Thus, the function f has a unique fixed point $\mathbf{z}_0$ that is the limit point of the subsquares $[a_0, b_0]^2, [a_1', b_1']^2, [a_2', b_2']^2, \ldots$. Finding the point $\mathbf{z}_0$ is equivalent to finding $\phi(0^1), \phi(0^2), \ldots$.

The basic construction of the function f_n follows the idea of the staircase construction of Hirsch *et al.* (1989) (the case $d = 2$). Indeed, the adversary argument of Hirsch *et al.* (1989) is almost identical to our argument for separating $PMLF_1$ from PF_1 by an oracle (Theorem 2.4). We first give a brief review of their construction of the function g. We omit the details that are irrelevant to our construction. For an integer $K > 4$, we first divide the square $[0, 1]^2$ into K^2 subsquares each of size $1/K \times 1/K$. We identify each subsquare whose lower left corner is the point $\langle s/K, t/K\rangle$ by the pair (s, t). These subsquares are divided into two parts. The outer most two layers, that is, squares (s, t) with s or t in $\{0, 1, K - 2, K - 1\}$, are called the *frame*, and the rest are called the *picture*. Each subsquare is labeled by an integer from 1 to 11. The labels of the subsquares in the frame are fixed:

(i) A subsquare (s, t) in the frame with $t \geq 2$ is labeled by 1.

(ii) A subsquare (s, t) in the frame with $t = 0$ is labeled by 11.

(iii) A subsquare (s, t) in the frame with $t = 1$ is labeled by 8 if $s \leq 1$, labeled by 9 if $s = 2$, and labeled by 10 if $s \geq 3$.

There is a *staircase* defined on the picture which is a sequence of subsquares of the shape of a staircase starting with the subsquare (2, 2) and ends with a subsquare (u, v) for some u and v between 2 and $K - 3$. The subsquares in the picture are labeled by integers from 1 to 6 as follows:

(i) The subsquares outside the staircase are labeled by 1.

(ii) The upward tracks of the staircase, except the endpoints, are labeled by 2.

(iii) The rightward tracks of the staircase, except the endpoints, are labeled by 3.

(iv) The subsquares that are the turning point from the upward tracks to the rightward tracks are labeled by 4.

(v) The subsquares that are the turning point from the rightward tracks to the upward tracks are labeled by 5.

(vi) The last subsquare in the staircase is labeled by 6.

It is best described by a picture. Figure 2 shows an example of a staircase.

The function g on each subsquare labeled by integer i, $1 \leq i \leq 11$, is defined by Template i. The reader is referred to Hirsch *et al.* (1989) for

1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1
1	1							1	1
1	1						6	1	1
1	1				4	3	5	1	1
1	1				2			1	1
1	1	4	3	3	5			1	1
1	1	2						1	1
8	8	9	10	10	10	10	10	10	10
11	11	11	11	11	11	11	11	11	11

FIG. 2. An example of a staircase with $K = 10$. The blank subsquares are those in the picture but not in the staircase and have label 1.

the precise definition of each template. Here, we only summarize the critical properties of the templates and the function g:

(a) The templates i, $1 \le i \le 11$, are defined in such a way that the two neighboring templates agree on their boundaries.

(b) For each point $\mathbf{z}$ on Template i, $1 \le i \le 11$, let $g(\mathbf{z}) = \mathbf{z} + g_{i,1}(\mathbf{z})$. Then, the direction of the vector $g_{i,1}(\mathbf{z})$ depends only on the relative position of $\mathbf{z}$ in the subsquare, and if $i \ne 6$ then the absolute value $|g_{i,1}(\mathbf{z})|$ is equal to c_g/K, where c_g is a constant with $0 < c_g < 1$.

(c) If $i \ne 6$, then the template i does not have a fixed point; and if $i = 6$, then the template i has a unique fixed point.

(d) g satisfies the Lipschitz condition with a Lipschitz constant M that is independent of K.

The above properties imply that the (unique) fixed point of g lies inside the subsquare labeled 6. The fixed point problem for a continuous function then is reduced to the combinatorial problem of searching for the subsquare that is labeled by 6. By an adversary argument similar to our argument in Theorem 2.4, it is proved in Hirsch *et al.* (1989) that this combinatorial search problem has an exponential lower bound.

Now we describe our basic construction. Assume that $\phi:\{0\}^* \to \{0, 1\}^*$ is a function in $PMLF_1$ that is not polynomial-time computable. Then, there exists a polynomial function p and, for each integer n, there is an increasing sequence $0^{p(n)} = y_0 < y_1 < \cdots < y_{k_n} = \phi(0^n)$ of strings of length $p(n)$ such that the mapping ψ from $(0^n, y_i)$ to (y_{i-1}, y_{i+1}) is polynomial-time computable. We now define, for each integer n, a function θ_n that maps each pair $(u, v) \in \{0, 1\}^{p(n)} \times \{0, 1\}^{p(n)}$ to an integer $i \in \{1, 2, \ldots, 6\}$ as

						6	
						2	
				4	3	5	
			4	5			
			2				
			2				
4	3	3	5				

FIG. 3. An example of function θ_n. All blank subsquares are labeled by 1.

$$\theta_n(u, v) = \begin{cases} 2, & \text{if } u = y_i \text{ and } y_{i-1} < v < y_i \text{ for some } i = 1, \ldots, k_n, \\ 3, & \text{if } y_{i-1} < u < y_i \text{ and } v = y_{i-1} \text{ for some } i = 1, \ldots, k_n, \\ 4, & \text{if } u = v = y_{i-1} \text{ for some } i = 1, \ldots, k_n, \\ 5, & \text{if } u = y_i \text{ and } v = y_{i-1} \text{ for some } i = 1, \ldots, k_n, \\ 6, & \text{if } u = v = y_{k_n}, \\ 1, & \text{otherwise.} \end{cases}$$

For instance, assume that $p(n) = 3$, and $y_0 = 000$, $y_1 = 011$, $y_2 = 100$, and $y_3 = y_{k_n} = 110$. Let us identify each string u of length $p(n)$ with the integer i_u, $0 \le i_u \le 2^{p(n)} - 1$, whose $p(n)$-bit binary representation is u. Then, function θ_n is a function defined on a standard chessboard, whose values are shown in Fig. 3.

Function θ_n thus defines a staircase. Since ψ is polynomial-time computable, it follows that the function $\theta(0^n, u, v) = \theta_n(u, v)$ is also polynomial-time computable.

Now we are ready to define the function f_n. Divide the square $[0, 1]^2$ into $2^{2p(n)}$ many subsquares each of size $2^{-p(n)} \times 2^{-p(n)}$. Identify each subsquare by integers (s, t) if its lower left corner is $\langle s \cdot 2^{-p(n)}, t \cdot 2^{-p(n)} \rangle$. Let s_n be the integer in $\{0, 1, \ldots, 2^{p(n)} - 1\}$ such that $\theta_n(s_n, s_n) = 6$, and let $a_n = s_n \cdot 2^{-p(n)}$ and $b_n = (s_n + 1)2^{-p(n)}$. The function f_n on $[0, 1]^2$ is a continuous function that encodes the labels $\theta_n(s, t)$ in the corresponding subsquares (s, t). That is, f_n on subsquare (s, t), with $\theta_n(s, t) = i$, is the same as the function g on Template i, with $K = 2^{p(n)}$.

The function f_n defined as above then has the property that its unique fixed point locates in the subsquare $[a_n, b_n]^2$, and so the problem of computing the fixed point of f_n is equivalent to the problem of computing the last subsquare (s_n, s_n) of the staircase defined by θ_n. However, this definition of f_n does

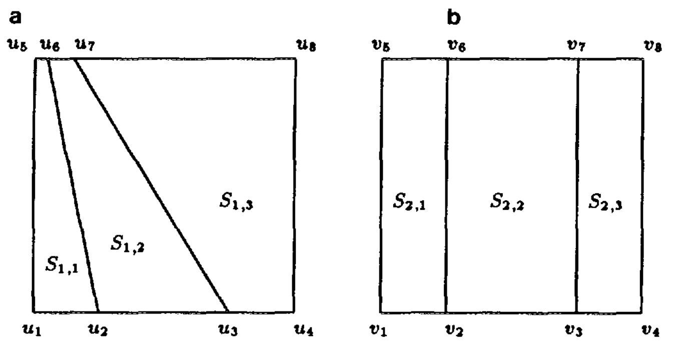

FIG. 4. The subareas of (a) S_1 and (b) S_2.

not satisfy all our needs. We note that, in step 2, the function f_{n+1} will be embedded into the subsquare $[a_n, b_n]^2$ of the function f_n. In order to make the final function f a continuous function, we need to make sure that the values of f_n at the boundary of $[a_n, b_n]^2$ are consistent with the values of f_{n+1} at the boundary of $[0, 1]^2$ (after applying the embedding mapping to it). To satisfy this additional requirement, we modify the function f_n on the subsquares (s, t) that are neighbors of the subsquare (s_n, s_n) as follows:

Case 1. Subsquare (s, t) is not the one right below the subsquare (s_n, s_n). Assume that $\theta_n(s, t) = i$. Let g_i be the function g defined on Template i (with $K = 2^{p(n)}$), and $g_{i,1}(\mathbf{z}) = g_i(\mathbf{z}) - \mathbf{z}$. For each $\mathbf{z}$ in the subsquare (s, t), let $d_{\mathbf{z}}$ be the L_∞-distance to the boundary of the subsquare (s_n, s_n),[3] and let $r_{\mathbf{z}} = 2^{p(n)}(1 - 2^{-p(n+1)})d_{\mathbf{z}} + 2^{-p(n+1)}$. We define $f_n(\mathbf{z}) = \mathbf{z} + r_{\mathbf{z}} \cdot g_{g,1}(\mathbf{z})$. That is, considering $f_{n,1}(\mathbf{z}) = f_n(\mathbf{z}) - \mathbf{z}$ as a vector, its direction is the same as the direction of $g_{i,1}(\mathbf{z})$ and $|f_{n,1}(\mathbf{z})| = r_{\mathbf{z}} \cdot |g_{i,1}(\mathbf{z})| = r_{\mathbf{z}} \cdot c_g \cdot 2^{-p(n)}$. Note that if $\mathbf{z}$ is at the boundary of the subsquare (s_n, s_n), then $r_{\mathbf{z}} = 2^{-p(n+1)}$, and $|f_{n,1}(\mathbf{z})| = c_g \cdot 2^{-(p(n)+p(n+1))}$.

Case 2. Subsquare (s, t) is the subsquare below the subsquare (s_n, s_n). That is, $s = s_n$ and $t = s_n - 1$. Let $c_n = (s_n - 1)2^{-p(n)}$. Then, the subsquare (s, t) is $[a_n, b_n] \times [c_n, a_n]$. We first define a piecewise linear transformation φ on $[a_n, b_n] \times [c_n, a_n]$. This piecewise linear transformation is complicated in the numerical form but is easy to illustrate in the picture form. In Fig. 4, we show two squares S_1 and S_2, both representing $[a_n, b_n] \times [c_n, a_n]$. Each square S_i, $i = 1, 2$, is divided into three subareas $S_{i,j}$, $j = 1, 2, 3$, whose corners are defined as

[3] The L_∞-distance between two points $\langle x_1, y_1\rangle$ and $\langle x_2, y_2\rangle$ is $\max\{|x_1 - x_2|, |y_1 - y_2|\}$.

$$\begin{array}{ll}
u_1 = v_1 = \langle a_n, c_n \rangle, & u_2 = v_2 = \langle a_n + (\tfrac{1}{4})2^{-p(n)}, c_n \rangle, \\
u_3 = v_3 = \langle a_n + (\tfrac{3}{4})2^{-p(n)}, c_n \rangle, & u_4 = v_4 = \langle b_n, c_n \rangle, \\
u_5 = v_5 = \langle a_n, a_n \rangle, & u_6 = \langle a_n + (\tfrac{1}{4})2^{-p(n)-p(n+1)}, a_n \rangle, \\
v_6 = \langle a_n + (\tfrac{1}{4})2^{-p(n)}, a_n \rangle, & u_7 = \langle a_n + (\tfrac{3}{4})2^{-p(n)-p(n+1)}, a_n \rangle, \\
v_7 = \langle a_n + (\tfrac{3}{4})1^{-p(n)}, a_n \rangle, & u_8 = v_8 = \langle b_n, a_n \rangle.
\end{array}$$

For each $j = 1, 2, 3$, we let φ_j be the linear function mapping $S_{1,j}$ to $S_{2,j}$, with $\varphi_j(u_k) = v_k$ for all $k = 1, \ldots, 8$. Let φ be the combination of φ_j, $j = 1, 2, 3$.

The function f_n on the subsquare $(s_n, s_n - 1)$ can be defined according to this mapping φ. Let $i = \theta_n(s, t)$. Let $g_i(\mathbf{z})$ be the function g defined on Template i, and let $g_{i,1}(\mathbf{z}) = g_i(\mathbf{z}) - \mathbf{z}$. Then, we define $f_n(\mathbf{z}) = \mathbf{z} + r_{\mathbf{z}} \cdot g_{i,1}(\varphi(\mathbf{z}))$, where $r_{\mathbf{z}}$ is defined as in Case 1. That is, the direction of the vector $f_{n,1}(\mathbf{z}) = f_n(\mathbf{z}) - \mathbf{z}$ is the same as that of $g_{i,1}(\varphi(\mathbf{z}))$ and $|f_{n,1}(\mathbf{z})| = r_{\mathbf{z}} \cdot c_g \cdot 2^{-p(n)}$.

Case 3. $(s, t) = (s_n, s_n)$. We leave f_n undefined on the subsquare (s_n, s_n) (except on its boundary).

The following observation on the function f_n on subsquare $(s_n, s_n - 1)$ will be used later.

CLAIM 1. *For each* $\mathbf{z} = \langle x, a_n \rangle$ *on the top boundary of* $[a_n, b_n] \times [c_n, a_n]$, *let* $\mathbf{z}' = \langle x', 0 \rangle$, *where* $x' = 2^{p(n)}(x - a_n)$. *Then,* $f_n(\mathbf{z}) - \mathbf{z}$ *and* $f_{n+1}(\mathbf{z}') - \mathbf{z}'$ *have the same direction.*

Proof. This claim depends on the precise definition of the templates. We note that subsquare $(s_n, s_n - 1)$ must have $\theta_n(s_n, s_n - 1) = i \in \{2, 5\}$. In either case, the direction of $g_i(\mathbf{z}) - \mathbf{z}$ is downward for all $\mathbf{z}$ lying in the rightmost quarter of the top boundary. (This follows from the definition of Templates 2 and 5.) So, after the transformation φ, $f_n(\mathbf{z}) - \mathbf{z}$ has the downward direction for all $\mathbf{z} = \langle x, a_n \rangle$, except when $a_n \leq x \leq a_n + 2^{-(p(n)+p(n+1))}$. On the line segment $[a_n, a_n + 2^{-(p(n)+p(n+1))}] \times \{a_n\}$, its direction is the same as that of the top of Template i on a square of size $2^{-(p(n)+p(n+1))}$.

To see that the direction of $f_n(\mathbf{z}) - \mathbf{z}$ agree with $f_{n+1}(\mathbf{z}') - \mathbf{z}'$, we observe that (i) all $f_{n+1}(\mathbf{z}') - \mathbf{z}'$ on the line segment $[2^{-p(n+1)}, 1] \times \{0\}$ have the downward direction (this follows from the definition of Templates 1, 3, and 5), and (ii) f_{n+1} on the square $[0, 2^{-p(n+1)}]^2$ is a Template 4, and the bottom of Template 4 agrees with the top of Template 2 and 5. Thus, Claim 1 is proven. ■

It is not hard to see that the absolute value of $f_{n,1}(\mathbf{z})$ can shrink from $g_1(\mathbf{z})$ by a factor at most $2^{-p(n+1)}$ and that the Lipschitz constant for f_n can increase by a factor at most $2^{p(n+1)}$. So, f_n has the following properties:

(a) f_n satisfies the Lipschitz condition with a Lipschitz constant $M \cdot 2^{p(n+1)}$, where M is a constant independent of n.

(b) For any $\mathbf{z}$ outside the subsquare (s_n, s_n), $c_g \cdot 2^{-(p(n)+p(n+1))} \leq |f_n(\mathbf{z}) - \mathbf{z}| \leq c_g \cdot 2^{-p(n)}$.

The above completes the basic construction for f_n on $[0, 1]^2 - [a_n, b_n]^2$. Next we define the function f as a combination of the functions f_n, applied with appropriate linear transformations. Let $[a, b]^2$ be a square of size $b - a = r$. Then, there is a unique linear bijection φ from $[0, 1]^2$ to $[a, b]^2$; namely, $\varphi(\langle x, y\rangle) = \langle a + rx, a + ry\rangle$.

Let $q(0) = 2$ and $q(n) = \sum_{i=1}^{n} p(n) + 2$ for $n > 0$. Without loss of generality, assume that $p(k) \geq k + 1$ for all $k \geq 1$ and $p(1) = 2$. We define f by stages.

Stage 0. We define f on $[0, 1]^2 - [\frac{1}{4}, \frac{3}{4}]^2$ to be the same as function g on the frame with size $K = 8$. Let $S_0 = [\frac{1}{4}, \frac{3}{4}]^2$. Note that S_0 is of size $2^{-q(0)} \times 2^{-q(0)}$.

Stage $n + 1$. Assume that in stage n, f has been defined on $[0, 1]^2$, except on a square $S_n = [a_n', b_n']^2$ of size $2^{-q(n)} \times 2^{-q(n)}$. Then, we define f on S_n to be the image of function f_{n+1} under the linear bijection φ_n that maps $[0, 1]^2$ to S_n. That is, $f(\varphi_n(\mathbf{z})) - \varphi_n(\mathbf{z}) = 2^{-q(n)}(f_{n+1}(\mathbf{z}) - \mathbf{z})$. Note that f_{n+1} is undefined on a subsquare of size $2^{-p(n+1)} \times 2^{-p(n+1)}$. The image of this square under φ_n is a square of size $2^{-q(n+1)} \times 2^{-q(n+1)}$. We let this square be S_{n+1}.

The above defined f on every $\mathbf{z}$ except the point $\mathbf{z}_0$ that is the intersection of all S_n, $n > 0$. We let $f(\mathbf{z}_0) = \mathbf{z}_0$. Note that the absolute value of $f(\mathbf{z}) - \mathbf{z}$ for $\mathbf{z} \in S_n$ converges to 0 as n tends to ∞; thus f is continuous at $\mathbf{z}_0$.

We verify that function f satisfies our need.

CLAIM 2. *f is well defined. That is, for each n, the values $f(\mathbf{z})$ defined in stage n and the values $f(\mathbf{z})$ defined in stage $n + 1$ agree on the boundary of square S_n.*

Proof. Let $\mathbf{z}$ be a point on the boundary of S_n. We let $f^{[j]}(\mathbf{z})$ denote the value of $f(\mathbf{z})$ defined in stage j, for $j = n, n + 1$. That is, $f^{[j]}(\mathbf{z}) = \mathbf{z} + 2^{-q(j-1)}(f_j(\varphi_{j-1}^{-1}(\mathbf{z})) - \varphi_{j-1}^{-1}(\mathbf{z}))$, where φ_{j-1} is the linear bijection from $[0, 1]^2$ to S_{j-1}, $j = n, n + 1$.

First, from the definition of f_n, we know that $|f_n(\mathbf{w}) - \mathbf{w}| = c_g \cdot 2^{-(p(n)+p(n+1))}$ for all $\mathbf{w}$ on the boundary of the subsquare with label 6. It follows that $|f^{[n]}(\mathbf{z}) - \mathbf{z}| = 2^{-q(n-1)} \cdot c_g \cdot 2^{-(p(n)+p(n+1))} = c_g \cdot 2^{-q(n+1)}$. Also, since $|f_{n+1}(\mathbf{w}) - \mathbf{w}| = c_g \cdot 2^{-p(n+1)}$ for all $\mathbf{w}$ on the boundary of $[0, 1]^2$, we have $|f^{[n+1]}(\mathbf{z}) - \mathbf{z}| = 2^{-q(n)} \cdot c_g \cdot 2^{-p(n+1)} = c_g \cdot 2^{-q(n+1)}$. Thus, the absolute value of $f^{[n]}(\mathbf{z})$ and $f^{[n+1]}(\mathbf{z})$ are equal.

It is left to verify that the two definitions of f agree on the direction of the vector $f(\mathbf{z}) - \mathbf{z}$. Note that on the left, right, and upper sides of S_n, the vectors $f^{[j]}(\mathbf{z}) - \mathbf{z}$, $j = n, n + 1$, all have the downward direction. This is clear for $j = n + 1$. For $j = n$, we note that in the definition of f_n, the left, right, and upper neighbors of the subsquare labeled by 6 must have label 1, and function f_n on Template 1 is always downward. Finally, we note that Claim 1 has established that the two definitions of f agree on the lower side of S_n. ■

CLAIM 3. *For all* $\mathbf{z} \in S_n - S_{n+1}$, $c_g \cdot 2^{-q(n+2)} \leq |f(\mathbf{z}) - \mathbf{z}| \leq c_g \cdot 2^{-q(n+1)}$.

Proof. This follows from Property (b) of f_{n+1}. ■

CLAIM 4. *For each* $\mathbf{z} \in [0, 1]^2$, $f(\mathbf{z}) \in [0, 1]^2$.

Proof. Note that $f(\mathbf{z})$ on the frame (defined in stage 0) are all within $[0, 1]^2$. For each $\mathbf{z} \in S_n - S_{n+1}$, we know that $f(\mathbf{z})$ lies either within S_n or, by Claim 3, within a distance $c_g \cdot 2^{-q(n+1)}$ from the boundary of S_n. Thus, $f(\mathbf{z})$ lies within $[0, 1]^2$, since $c_g < 1$. ■

CLAIM 5. *f has a polynomial modulus on* $[0, 1]^2$.

Proof. Assume that $\mathbf{z}_1, \mathbf{z}_2 \in S_n - S_{n+1}$. Let $\mathbf{w}_j = \varphi_n^{-1}(\mathbf{z}_j)$ for $j = 1, 2$. Recall that f_{n+1} satisfies the Lipschitz condition with the Lipschitz constant $M2^{p(n+2)}$. It follows then that

$$\begin{aligned}|f(\mathbf{z}_1) - f(\mathbf{z}_2)| &\leq |\mathbf{z}_1 - \mathbf{z}_2| + 2^{-q(n)}(|\mathbf{w}_1 - \mathbf{w}_2| + |f_{n+1}(\mathbf{w}_1) - f_{n+1}(\mathbf{w}_2)|)\\ &\leq 2|\mathbf{z}_1 - \mathbf{z}_2| + 2^{-q(n)} \cdot M \cdot 2^{p(n+2)} \cdot |\mathbf{w}_1 - \mathbf{w}_2|\\ &= (M2^{p(n+2)} + 2)|\mathbf{z}_1 - \mathbf{z}_2|.\end{aligned}$$

Define $p_1(k) = p(k + 2) + k + \lceil \log M \rceil + 2$. We claim that p_1 is a polynomial modulus function for f. Assume that $|\mathbf{z}_1 - \mathbf{z}_2| \leq 2^{-p_1(k)}$. We consider four cases. First, if $\mathbf{z}_1, \mathbf{z}_2 \in S_k$ then, by Claim 3, $|f(\mathbf{z}_1) - f(\mathbf{z}_2)| \leq |\mathbf{z}_1 - \mathbf{z}_2| + 2 \cdot c_g \cdot 2^{-q(k+1)} \leq 4 \cdot 2^{-q(k)} \leq 2^{-(k+1)}$, since $\mathbf{z}_1, \mathbf{z}_2 \in S_k$ implies that $|\mathbf{z}_1 - \mathbf{z}_2| \leq 2 \cdot 2^{-q(k)}$.

Next, assume that $\mathbf{z}_1, \mathbf{z}_2 \in S_n - S_{n+1}$ for some $n < k$. Then $|f(\mathbf{z}_1) - f(\mathbf{z}_2)| \leq (M \cdot 2^{p(n+2)} + 2)|\mathbf{z}_1 - \mathbf{z}_2| \leq 2^{-(k+1)}$ by the above observation. Furthermore, if $|\mathbf{z}_1 - \mathbf{z}_2| = r \cdot 2^{-p_1(k)}$ for some $r \in [0, 1]$, then $|f(\mathbf{z}_1) - f(\mathbf{z}_2)| \leq r \cdot 2^{-(k+1)}$.

For the third case, assume that $\mathbf{z}_1 \in S_n - S_{n+1}$ and $\mathbf{z}_2 \in S_m - S_{m+1}$, with $n < m < k$. Then, for each j, $n < j \leq m$, we may find $\mathbf{z}^{(j)}$ in the boundary of S_j such that $\sum_{j=n}^{m} |\mathbf{z}^{(j)} - \mathbf{z}^{(j+1)}| = |\mathbf{z}_1 - \mathbf{z}_2|$, where $\mathbf{z}^{(n)} = \mathbf{z}_1$ and $\mathbf{z}^{(m+1)} = \mathbf{z}_2$. For each j, $n \leq j \leq m$, let $r_j = |\mathbf{z}^{(j)} - \mathbf{z}^{(j+1)}| \cdot |\mathbf{z}_1 - \mathbf{z}_2|^{-1}$. It now follows from the above case that $|f(\mathbf{z}^{(j)}) - f(\mathbf{z}^{(j+1)})| \leq r_j \cdot 2^{-(k+1)}$, and so $|f(\mathbf{z}_1) - f(\mathbf{z}_2)| \leq 2^{-(k+1)}$.

Finally, if $\mathbf{z}_1 \in S_n - S_{n+1}$ and $\mathbf{z}_2 \in S_k$, then we can find a point $\mathbf{z}_3$ on the boundary of S_k such that $|\mathbf{z}_1 - \mathbf{z}_3| + |\mathbf{z}_3 - \mathbf{z}_2| = |\mathbf{z}_1 - \mathbf{z}_2|$. By the first and third cases above, we have $|f(\mathbf{z}_1) - f(\mathbf{z}_2)| \le |f(\mathbf{z}_1) - f(\mathbf{z}_3)| + |f(\mathbf{z}_3) - f(\mathbf{z}_2)| \le 2 \cdot 2^{-(k+1)} = 2^{-k}$. ■

CLAIM 6. *f is polynomial-time computable.*

Proof. By Claim 5, we only need to check that for each dyadic point $\mathbf{d}$, we can compute, in time polynomial in n, an approximate value $\mathbf{e}$ to $f(\mathbf{d})$ within error 2^{-n}. This amounts to (i) find the integer k such that $\mathbf{d} \in S_{k-1} - S_k$ or $\mathbf{d} \in S_{k-1}$ and $k > n$, and (ii) if $\mathbf{d} \in S_{k-1} - S_k$, find an approximation to $f_k(\mathbf{w})$, where $\mathbf{w} = \varphi_{k-1}^{-1}(\mathbf{z})$.

For the problem (i), let us assume that we have already found $S_{k-1} = [a'_{k-1}, b'_{k-1}]^2$, and determined that $\mathbf{d} \in S_{k-1}$. Now, let $\mathbf{w} = \varphi_{k-1}^{-1}(\mathbf{d})$ and find the subsquare $[s \cdot 2^{-p(k)}, t \cdot 2^{-p(k)}]$ that contains $\mathbf{w}$. Determine whether $\theta_k(s, t) = 6$. If so, then we have found S_k (namely, $a'_k = a'_{k-1} + s \cdot 2^{-q(k)}$ and $S_k = [a'_k, a'_k + 2^{-q(k)}]^2$). Otherwise, $\mathbf{d} \in S_{k-1} - S_k$. The problem (ii) can be solved then easily from $\theta_k(s, t)$. ■

CLAIM 7. *$f_1(\mathbf{z}) = f(\mathbf{z}) - \mathbf{z}$ has a polynomial inverse modulus at zeros.*

Proof. The only zero of f_1 is $\mathbf{z}_0$. If $|\mathbf{z} - \mathbf{z}_0| \ge 2^{-q(n)}$, then $\mathbf{z} \notin S_n$, and it follows that $|f_1(\mathbf{z})| \ge c_g \cdot 2^{-q(n+1)}$. ■

CLAIM 8. *$\mathbf{z}_0$ is the unique fixed point of f, and $\mathbf{z}_0$ is not polynomial-time computable.*

Proof. Assume that the lower left corner of S_n is $\langle a'_n, a'_n \rangle$. Then, we have $a'_{n+1} = a'_n + \phi(0^{n+1}) \cdot 2^{-q(n+1)}$. It follows that $x_0 = \frac{1}{4} + \sum_{n=1}^{\infty} \phi(0^n) \cdot 2^{-q(n)}$, where $\mathbf{z}_0 = \langle x_0, x_0 \rangle$. In other words, the value $\phi(0^n)$ is exactly the subsequence of the binary expansion of x_0 from the $(q(n-1)+1)$th bit to the $q(n)$th bit in the fractional part. So, this number x_0 is not polynomial-time computable. To be more precise, if x_0 were polynomial-time computable, then we would be able to obtain, in time polynomial in n, a diadic rational $d \in [0, 1]$ having $q(n)$ bits in the fractional part such that $|d - x_0| \le 2^{-q(n)}$. Let u be the last $p(n)$ bits of d. Then $\phi(0^n)$ would be equal to either u (in case $d \le x_0$) or u', the predecessor of u (in case $d > x_0$). (The predecessor of $1^{p(n)}$ is $0^{p(n)}$.) We could determine which case it is by running $\psi(0^n, u)$ and $\psi(0^n, u')$ in polynomial time. ■

5. COMPLEXITY OF THE INTERSECTION POINTS

Let f and g be two polynomial-time computable functions from $[0, 1]$ to $[0, 1]^2$ with the properties that $f(0)$ lies on the left side of the unit square, $f(1)$ lies on the right side of the unit square, $g(0)$ lies on the upper side of the unit square, and $g(1)$ lies on the lower side of the unit square. We are concerned with the complexity of the intersection points of the two curves defined by f and g. Similar to the case of fixed points, the intersection points of two polynomial-time computable curves on $[0, 1]^2$ could have arbitrarily high complexity if the two curves are allowed to get arbitrarily close to each other away from the intersection points. We restrict ourselves to functions f and g for which any approximate intersection point is a good approximation to an exact intersection point. We say (s_0, t_0) *defines an intersection point* of f and g if $f(s_0) = g(t_0)$. We say two functions f and g from $[0, 1]$ to $[0, 1]^2$ are *polynomially distinguishable away from the intersection points* (or, simple *well-conditioned*) if there exists a polynomial function q such that if $|s - s_0| \geq 2^{-n}$ and $|t - t_0| \geq 2^{-n}$ for all pairs (s_0, t_0) that define intersection points, then $|f(s) - g(x)| \geq 2^{-q(n)}$ and $|f(x) - g(t)| \geq 2^{-q(n)}$ for all $x \in [0, 1]$. The complexity of the intersection points of well-conditioned, polynomial-time computable functions f and g have an upper bound NP.

PROPOSITION 5.1. *Assume that f and g are two well-conditioned, polynomial-time computable functions mapping $[0, 1]$ to $[0, 1]^2$ such that $f(0) = \langle 0, a\rangle$, $f(1) = \langle 1, b\rangle$, $g(0) = \langle c, 1\rangle$, and $g(1) = \langle d, 0\rangle$ for some numbers a, b, c, $d \in [0, 1]$. If $P = NP$, then all isolated intersection points of f and g are polynomial-time computable.*

We do not know whether this upper bound can be improved. It appears that Sperner's lemma does not apply here, since functions f and g do not map the unit square to itself.

In the special case that f is a graph of a polynomial-time computable function $h: [0, 1] \to [0, 1]$, the intersection points are provably polynomial-time computable.

THEOREM 5.2. *Assume that f and g satisfy the assumption of Proposition 5.1. Further assume that there is a polynomial-time computable $h: [0, 1] \to [0, 1]$ such that $f(t) = \langle t, h(t)\rangle$. Then, at least one intersection point of f and g is polynomial-time computable.*

Proof (Sketch). Let $U = \{\langle x, y\rangle : 0 \leq x \leq 1, y \geq h(x)\}$, and let $L = \{\langle x, y\rangle : 0 \leq x \leq 1, y \geq h(x)\}$. Then, $g(0) \in U$ and $g(1) \in L$. For any $t \in [0, 1]$ with $g(t) = \langle x, y\rangle$, we can determine, in time polynomial in n, whether $g(t) \in U$, or $g(t) \in L$, or $|g(t) - f(x)| \leq 2^{-n}$, by comparing y with $h(x)$. Thus, a binary search finds, in time polynomial in n, t and s in $[0, 1]$ such that $|g(t) - f(s)| \leq 2^{-n}$. Since f and g are well-conditioned, $g(t)$ is a good approximation to an intersection point. ■

In the following, we show that in general, the intersection points are hard to compute.

THEOREM 5.3. *Assume that $PMLF_1 \not\subseteq PF_1$. Then, there exist two well-conditioned, polynomial-time computable, one-to-one functions f, $g: [0, 1] \to [0, 1]^2$ such that*

(i) $f(0) = \langle 0, 0\rangle, f(1) = \langle 1, 1\rangle, g(0) = \langle 0, 1\rangle, g(1) = \langle 1, 0\rangle$,

(ii) *f and g have a unique intersection point* $f(s_0) = g(t_0) = \mathbf{z}_0$ *in* $[0, 1]^2$, *and*

(iii) $\mathbf{z}_0$ *is not polynomial-time computable.*

Proof. The proof, similar to the proof of Theorem 4.3, consists of two steps. Assume that $\phi \in PMLF_1 - PF_1$. First, for each integer n, we describe two functions f_n, g_n such that the intersection point of f_n and g_n encodes the value of $\phi(0^n)$. Then, in the second step, we combine functions f_n into a function f and functions g_n into a function g.

Basic Construction. Assume that $\phi: \{0\}^* \to \{0, 1\}^*$ is a function in $PMLF_1$ that is not polynomial-time computable. Then, there exists a polynomial function p and, for each integer n, there is an increasing sequence $0^{p(n)} = y_0 < y_1 < \cdots < y_{k_n} = \phi(0^n)$ of strings of length $p(n)$ such that the mapping ψ from $(0^n, y_i)$ to (y_{i-1}, y_{i+1}) is polynomial-time computable. Without loss of generality, we assume that $p(n) \geq n$ for all $n \geq 1$.

We let

$$t_i = \tfrac{1}{4} + i \cdot 2^{-(p(n)+1)}, \qquad \mathbf{a}_i = \langle t_i, \tfrac{3}{4}\rangle, \qquad \mathbf{b}_i = \langle t_i, \tfrac{1}{4}\rangle,$$

for $i = 0, 1, \ldots, 2^{p(n)}$. For each i, $0 \leq i \leq 2^{p(n)} - 1$, let w_i be the ith string in $\{0, 1\}^{p(n)}$ (i.e., w_i is the $p(n)$-bit binary representation of integer i). The functions f_n and g_n are piecewise linear functions defined as follows. (In the following, we let $\overline{\mathbf{z}_1 \mathbf{z}_2}$ denote the line segment connecting point $\mathbf{z}_1$ to point $\mathbf{z}_2$.)

(1) On interval $[0, t_0]$, f_n linearly maps the interval $[0, t_0]$ to the line segment $\overline{\langle 0, 0\rangle\, \mathbf{b}_0}$; g_n linearly maps $[0, t_0]$ to the line segment $\overline{\langle 0, 1\rangle\, \mathbf{a}_0}$.

(2) For each i, $0 \leq i \leq 2^{p(n)} - 1$, if $w_i \neq y_j$ for any $j = 0, 1, \ldots, k_n$, then f_n and g_n on $[t_i, t_{i+1}]$ are two linear mappings: f_n maps $[t_i, t_{i+1}]$ linearly to the line segment $\overline{\mathbf{a}_i\, \mathbf{a}_{i+1}}$ and g_n maps $[t_i, t_{i+1}]$ linearly to the line segment $\overline{\mathbf{b}_i\, \mathbf{b}_{i+1}}$.

(3) For $i = 0$, we have $w_i = y_0$. Let $y_1 = w_m$. Functions f_n and g_n are piecewise linear on $[t_0, t_1]$. We divide $[t_0, t_1]$ into five subintervals of equal length, and we let f_n map these subintervals to five consecutive line segments defined by the following six points:

$$\begin{array}{lll} \mathbf{b}_0 = \langle t_0, \tfrac{1}{4}\rangle, & \langle t_0, \tfrac{1}{8}\rangle, & \langle t_m + (\tfrac{1}{5}) \cdot 2^{-(p(n)+1)}, \tfrac{1}{8}\rangle, \\ \langle t_m + (\tfrac{1}{5}) \cdot 2^{-(p(n)+1)}, \tfrac{5}{8}\rangle, & \langle t_1, \tfrac{5}{8}\rangle, & \mathbf{a}_1 = \langle t_1, \tfrac{3}{4}\rangle. \end{array}$$

g_n maps $[t_0, t_1]$ into three consecutive line segment defined by the following four points:

$$\mathbf{a}_0 = \langle t_0, \tfrac{3}{4}\rangle, \qquad \langle t_0 + (\tfrac{2}{5})\cdot 2^{-(p(n)+1)}, \tfrac{3}{4}\rangle,$$
$$\langle t_0 + (\tfrac{2}{5})\cdot 2^{-(p(n)+1)}, \tfrac{1}{4}\rangle, \qquad \mathbf{b}_1 = \langle t_1, \tfrac{1}{4}\rangle.$$

(4) For any integer i, $0 < i < 2^{p(n)}$, if $w_i = y_j$ for some j, $0 < j < k_n$, then we can use function ψ to find y_{j-1} and y_{j+1}. Assume that $y_{j-1} = w_l$ and $y_{j-1} = w_m$. Function f_n linearly maps $[t_i, t_{i+1}]$ into six consecutive line segments defined by the following seven points:

$$\mathbf{a}_i = \langle t_i, \tfrac{3}{4}\rangle, \qquad \langle t_i + (\tfrac{2}{5})\cdot 2^{-(p(n)+1)}, \tfrac{3}{4}\rangle, \qquad \langle t_i + (\tfrac{2}{5})\cdot 2^{-(p(n)+1)}, \tfrac{1}{8}\rangle,$$
$$\langle t_m + (\tfrac{1}{5})\cdot 2^{-(p(n)-1)}, \tfrac{1}{8}\rangle, \qquad \langle t_m + (\tfrac{1}{5})\cdot 2^{-(p(n)+1)}, \tfrac{5}{8}\rangle, \qquad \langle t_{i+1}, \tfrac{5}{8}\rangle,$$
$$\mathbf{a}_{i+1} = \langle t_{i+1}, \tfrac{3}{4}\rangle.$$

Function g_n linearly maps $[t_i, t_{i+1}]$ into six consecutive line segments defined by the following seven points:

$$\mathbf{b}_i = \langle t_i, \tfrac{1}{4}\rangle, \qquad \langle t_i + (\tfrac{3}{8}), \qquad \langle t_l + (\tfrac{4}{5})\cdot 2^{-(p(n)+1)}, \tfrac{3}{8}\rangle,$$
$$\langle t_l + (\tfrac{4}{5})\cdot 2^{-(p(n)+1)}, \tfrac{7}{8}\rangle, \qquad \langle t_i + (\tfrac{3}{5})\cdot 2^{-(p(n)+1)}, \tfrac{7}{8}\rangle, \qquad \langle t_i + (\tfrac{3}{5})\cdot 2^{-(p(n)+1)}, \tfrac{1}{4}\rangle,$$
$$\mathbf{b}_{i+1} = \langle t_{i-1}, \tfrac{1}{4}\rangle.$$

(5) For integer i with $w_i = y_{k_n}$, let $w_l = y_{k_n-1}$. Let $u_n = (t_i + t_{i+1})/2$, $u'_n = t_{i+1}$, $v_n = \frac{1}{2}$, and $v'_n = \frac{1}{2} + 2^{\,(p(n)+2)}$. The function f_n maps $[t_i, (t_i + t_{i-1})/2]$ piecewise linearly to three consecutive line segments defined by the following four points:

$$\mathbf{a}_i = \langle t_i, \tfrac{3}{4}\rangle, \qquad \langle t_i + (\tfrac{2}{5})\cdot 2^{-(p(n)+1)}, \tfrac{3}{4}\rangle, \qquad \langle t_i + (\tfrac{2}{5})\cdot 2^{-(p(n)+1)}, v_n\rangle, \qquad \langle u_n, v_n\rangle.$$

The function f_n also maps $[(t_i + t_{i+1})/2, t_{i-1}]$ linearly to two consecutive line segments defined by the following three points:

$$\langle u_n, v_n\rangle, \qquad \langle u'_n, v'_n\rangle, \qquad \mathbf{a}_{i+1}.$$

Function g_n maps $[t_i, (t_i + t_{i+1})/2]$ piecewise linearly to five consecutive line segments defined by the following six points:

$$\mathbf{b}_i = \langle t_i, \tfrac{1}{4}\rangle, \qquad \langle t_i, \tfrac{3}{8}\rangle, \qquad \langle t_l + (\tfrac{4}{5})\cdot 2^{-(p(n)-1)}, \tfrac{3}{8}\rangle,$$
$$\langle t_l + (\tfrac{4}{5})\cdot 2^{-(p(n)+1)}, \tfrac{7}{8}\rangle, \qquad \langle u_n, \tfrac{7}{8}\rangle, \qquad \langle u_n, v'_n\rangle.$$

g_n also maps $[(t_i + t_{i+1})/2, t_{i+1}]$ into two consecutive line segments defined by the following three points:

$$\langle u_n, v'_n\rangle, \qquad \langle u'_n, v_n\rangle, \qquad \mathbf{b}_{i+1}.$$

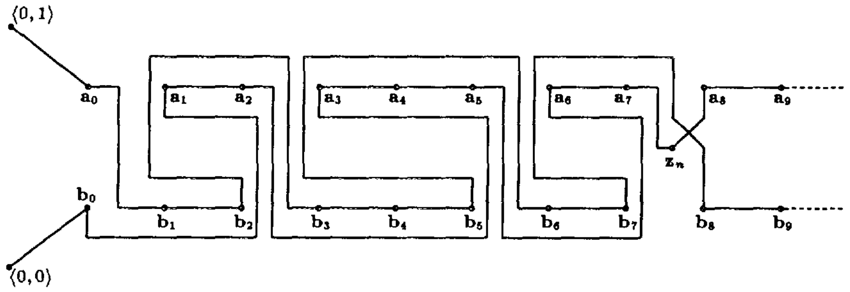

FIG. 5. The functions f_n and g_n, corresponding to $y_0 = w_0$, $y_1 = w_2$, $y_2 = w_5$, and $y_3 = y_{k_n} = w_7$. The point $\mathbf{z}_n$ is $\langle u_n, v_n\rangle$.

(6) Function f_n maps the interval $[\frac{3}{4}, 1]$ linearly to the line segment $\overline{\mathbf{a}_{2^{p(n)}}\langle 1, 1\rangle}$ and g_n maps $[\frac{3}{4}, 1]$ linearly to the line segment $\overline{b_{2^{p(n)}}\langle 1, 0\rangle}$.

We show in Fig. 5 functions f_n and g_n for the sequence $y_0 = w_0$, $y_1 = w_2$, $y_2 = w_5$, and $y_3 = y_{k_n} = w_7$.

For each integer $n \geq 1$, let i_n be the integer such that $y_{k_n} = w_{i_n}$. Define $a_n = (t_{i_n} + t_{i_n+1})/2$ and $b_n = (t_{i_n} + 3t_{i_n+1})/4$. Also let T_n be the square $[u_n, u_n'] \times [v_n, v_n']$. We note that function f_n maps $[0, a_n]$ to a curve from $\langle 0, 0\rangle$ to the point $\langle u_n, v_n\rangle$ and maps $[b_n, 1]$ to a curve from the point $\langle u_n', v_n'\rangle$ to $\langle 1, 1\rangle$. Function g_n maps $[0, a_n]$ to a curve from $\langle 0, 1\rangle$ to the point $\langle u_n, v_n'\rangle$, and maps $[b_n, 1]$ to a curve from $\langle u_n', v_n\rangle$ to $\langle 1, 0\rangle$. In addition, functions f_n and g_n satisfy the following properties:

(i) f_n and g_n are one-to-one functions on $[0, 1]$.

(ii) f_n and g_n satisfy the Lipschitz condition with the Lipschitz constant $L = 12 \cdot 2^{p(n)}$.

(iii) For any two points $\mathbf{z}_1$ on the curve defined by f_n and $\mathbf{z}_2$ on the curve defined by g_n, if $\mathbf{z}_1 \notin T_n$ or $\mathbf{z}_2 \notin T_n$, then $|\mathbf{z}_1 - \mathbf{z}_2| \geq (\frac{1}{5})2^{-(p(n)+1)}$. (In particular, f_n and g_n have a unique intersection point in T_n.)

(iv) The point u_n encodes the value of $\phi(0^n)$. More precisely, $u_n = t_{i_n} + 2^{-(p(n)+2)} = \frac{1}{4} + (2\phi(0^n) + 1)2^{-(p(n)+2)}$.

Combination. For any square $[a, b] \times [c, d]$, with $r = b - a = d - c$, there is a unique linear bijection φ from $[0, 1]^2$ to $[a, b] \times [c, d]$; namely, $\varphi(\langle x, y\rangle) = \langle a + rx, c + ry\rangle$.

We define functions f and g by stages. We are going to define a sequence of intervals $[c_n, d_n]$ such that $c_n < c_{n+1} < d_{n+1} < d_n$. At stage n, we define functions f and g on $[0, c_n] \cup [d_n, 1]$.

Stage 1. Let $c_1 = a_1$, $d_1 = b_1$. Let f on $[0, c_1] \cup [d_1, 1]$ be equal to f_1

on these two subintervals. Also let g on $[0, c_1] \cup [d_1, 1]$ be equal to g_1 on these two subintervals. Note that $f(c_1) = \langle u_1, v_1\rangle$, $f(d_1) = \langle u_1', v_1'\rangle$, $g(c_1) = \langle u_1, v_1'\rangle$, and $g(d_1) = \langle u_1', v_1\rangle$. We let $\alpha_1 = u_1$, $\alpha_1' = u_1'$, $\beta_1 = v_1$, $\beta_1' = v_1'$, and let $S_1 = [\alpha_1, \alpha_1'] \times [\beta_1, \beta_1']$.

Stage $n + 1$. Assume that in stage n, a square $S_n = [\alpha_n, \alpha_n'] \times [\beta_n, \beta_n']$ has been defined, and functions f and g have been defined on $[0, c_n] \cup [d_n, 1]$ with $f(c_n) = \langle \alpha_n, \beta_n\rangle$, $f(d_n) = \langle \alpha_n', \beta_n'\rangle$, $g(c_n) = \langle \alpha_n, \beta_n'\rangle$, and $g(d_n) = \langle \alpha_n', \beta_n\rangle$. Let ϑ_n be the linear transformation from $[c_n, d_n]$ to $[0, 1]$; that is, $\vartheta_n(t) = (t - c_n)/(d_n - c_n)$. We define $c_{n+1} = \vartheta_n^{-1}(a_{n+1})$ and $d_{n+1} = \vartheta_n^{-1}(b_{n+1})$. Let φ_n be the unique linear bijection from $[0, 1]^2$ to S_n. For each $t \in [c_n, c_{n+1}] \cup [d_{n+1}, d_n]$, we define $f(t) = \varphi_n(f_{n+1}(\vartheta_n(t)))$ and $g(t) = \varphi_n(g_{n+1}(\vartheta_n(t)))$.

Let $r_n = \alpha_n' - \alpha_n = \beta_n' - \beta_n$, and

$$\alpha_{n+1} = \alpha_n + r_n \cdot u_{n+1}, \qquad \alpha_{n+1}' = \alpha_n + r_n \cdot u_{n+1}',$$

$$\beta_{n+1} = \beta_n + r_n \cdot v_{n+1}, \qquad \beta_{n+1}' = \beta_n + r_n \cdot v_{n+1}'.$$

Then, we have $f(c_{n+1}) = \langle \alpha_{n+1}, \beta_{n+1}\rangle$, $f(d_{n+1}) = \langle \alpha_{n+1}', \beta_{n+1}'\rangle$, $g(c_{n+1}) = \langle \alpha_{n+1}, \beta_{n+1}'\rangle$, and $g(d_{n+1}) = \langle \alpha_{n+1}', \beta_{n+1}\rangle$. We define $S_{n+1} = [\alpha_{n+1}, \alpha_{n+1}'] \times [\beta_{n+1}, \beta_{n+1}']$. This completes the construction at Stage $n + 1$.

Let $t_0 = \lim_{n\to\infty} c_n$. The above completes the construction of f and g on $[0, 1] - \{t_0\}$. Let $\alpha_0 = \lim_{n\to\infty} \alpha_n$ and $\beta_0 = \lim_{n\to\infty} \beta_n$. We define $f(t_0) = g(t_0) = \mathbf{z}_0 = \langle \alpha_0, \beta_0\rangle$. This completes the construction of f and g.

It is easy to see by inspection that f and g are continuous functions mapping $[0, 1]$ to $[0, 1]^2$. We verify that they satisfy our requirements. Let $q_2(n) = \sum_{i=1}^n p(i) + 2n$ and $q_3(n) = \sum_{i=1}^n p(i) + 3n$. Note that the linterval $[c_n, d_n]$ is of length $2^{-q_3(n)}$ and the square S_n is of size $2^{-q_2(n)} \times 2^{-q_2(n)}$.

CLAIM 1. *f and g have polynomial modulus functions.*

Proof. We claim that $p_1(n) = p(n + 1) + 2n + 5$ is a modulus function for functions f and g. To see this, we assume that $t_1, t_2 \in [0, 1]$ and $|t_1 - t_2| \le 2^{-p_1(k)}$. We observe that if $t_1, t_2 \in [c_k, d_k]$, then both $f(t_1)$ and $f(t_2)$ lie in the square S_k, and so $|f(t_1) - f(t_2)| \le 2^{-q_2(k)} \le 2^{-k}$.

Next, following the same argument as in the proof of Theorem 4.3, we only need to prove that if $t_1, t_2 \in [c_n, d_n] - [c_{n+1}, d_{n+1}]$ with $|t_1 - t_2| \le r \cdot 2^{-p_1(k)}$ for some $n < k$ and some $0 \le r \le 1$, then $|f(t_1) - f(t_2)| \le r \cdot 2^{-(k+1)}$. In this case, we know that $f(t_j) = \varphi_n(f_{n+1}(\vartheta_n(t_j)))$ for $j = 1, 2$, and we recall that f_{n+1} satisfies the Lipschitz condition with the Lipschitz constant $L_{n+1} = 2^{p(n+1)+4}$. Thus,

$$\begin{aligned}|f(t_1) - f(t_2)| &= 2^{-q_2(n)} \cdot |f_{n+1}(\vartheta_n(t_1)) - f_{n+1}(\vartheta_n(t_2))| \\ &\leq 2^{-q_2(n)} \cdot 2^{p(n+1)+4} \cdot 2^{q_3(n)} \cdot r \cdot 2^{-p_1(k)} \\ &\leq 2^{p(n+1)+n+4} \cdot r \cdot 2^{-p_1(k)} \leq r \cdot 2^{-(k+1)}.\end{aligned}$$

The proof for function g is similar. ■

CLAIM 2. *f and g are polynomial-time computable.*

Proof. The proof is similar to that of Theorem 4.3. Namely, we need to compute, in time polynomial in k, for a given dyadic rational $d \in [0, 1]$: (i) an integer $n \leq k$ such that $d \in [c_n, d_n] - [c_{n+}, d_{n+1}]$ (or that $d \in [c_k, d_k]$); and (ii) an approximation to $f_{n+1}(\vartheta_n(d))$ within distance 2^{-k}.

For problem (i), let us assume that we already know that $d \in [c_n, d_n]$. Now, we compute $\vartheta_n(d)$ and find the corresponding t_i such that $t_i \leq \vartheta_n(d) \leq t_{i+1}$ (where t_i is the value $\frac{1}{4} + i \cdot 2^{-(p(n)+1)}$ defined in the basic construction step). Then, we verify whether $y_{k_n} = w_{i_n}$. If so, then we have found $[c_{n+1}, d_{n+1}]$; and if not, then we know $d \notin [c_{n+1}, d_{n+1}]$. In the latter case, the corresponding point $f_{n+1}(\vartheta_n(d))$ is easy to find from the basic construction. The proof for function g is similar. ■

CLAIM 3. *f and g are well-conditioned.*

Proof. Assume that $|s - t_0|$ is greater than or equal to 2^{-k}. This implies that $s \notin [c_k, d_k]$. So, $s \in [c_n, d_n] - [c_{n-1}, d_{n+1}]$ for some $n < k$. We check, in the following, that $|f(s) - g(t)| \geq 2^{-(q_2(k+1)+4)}$ for all $t \in [0, 1]$. The proof for $|g(s) - f(t)| \geq 2^{-(q_2(k+1)+4)}$ is symmetric and is omitted.

Case 1. $t \in [c_n, d_n] - [c_{n+1}, d_{n+1}]$. Then, we know that $|f(s) - g(t)|$ is equal to $2^{-q_2(n)}$ times $|f_{n+1}(s') - g_{n+1}(t')|$ for some $s', t' \in [0, 1] - [a_{n+1}, b_{n+1}]$. It follows from property (iii) of f_{n+1} and g_{n+1} that $|f(s) - g(t)| \geq (\frac{1}{5})2^{-q_2(n+1)} \geq 2^{-(q_2(k+1)+4)}$.

Case 2. $t \in [c_m, d_m] - [c_{m+1}, d_{m+1}]$ with $n < m - 1 \leq k - 1$. Then, $g(t)$ lies within the square S_{n+2} and $f(s)$ is outside S_{n+1}. We note that, in the construction of functions f_{n+1} and g_{n+1}, the square T_{n+1} has a distance at least $\frac{1}{4}$ from the boundary of $[0, 1]^2$. Thus, the square S_{n+2} has a distance at least $2^{-(q_2(n+1)+2)}$ from the boundary of the square S_{n+1}. This implies that $g(t)$, which is inside S_{n+2} has a distance at least $2^{-(q_2(n+1)+2)}$ from $f(s)$, which is outside S_{n+1}. Thus, $|f(s) - g(t)| \geq 2^{-(q_2(n+1)+2)} \geq 2^{-(q_2(k+1)+4)}$.

Case 3. $t \in [c_{n+1}, d_{n+1}] - [c_{n+2}, d_{n+2}]$. First, assume that $c_n \leq s \leq c_{n+1}$. We know then that $f(s)$ lies outside the square S_{n+1} and that $g(t)$ lies inside S_{n+1}. Now, the distance between $f(s)$ and the boundary of S_{n+1} is either greater than or equal to $2^{-(q_2(n+2)+4)}$ or less than that. In the former case, it is clear that $|f(s) - g(t)| \geq 2^{-(q_2(n+2)+4)} \geq 2^{-(q_2(k+1)+4)}$, since $g(t)$ is inside S_{n+1}. In the latter case, we know from the definition of function f_{n+1} that $|f(s) - f(c_{n+1})| \leq 2^{-(q_2(n+2)+4)}$. (Note that any line segment of f_{n+1} outside the square T_{n+1} is of length at least $(\frac{1}{5})2^{-(p(n+1)+1)}$, and so $f(s)$ and $f(c_{n+1})$ are in the same line segment.) Now, from property (iii) of f_{n+2}, we get $|f(c_{n+1}) - g(t)| \geq 2^{-q_2(n+1)} \cdot \frac{1}{5} \cdot 2^{-(p(n+2)+1)} \geq (\frac{1}{5})2^{-q_2(n+2)}$. It follows that

$$|f(s) - g(t)| \geq (\tfrac{1}{5})2^{-q_2(n+2)} - 2^{-(q_2(n+2)+4)} \geq 2^{-(q_2(n+2)+4)} \geq 2^{-(q_2(k+1)+4)}.$$

The case of $d_{n+1} \leq s \leq d_n$ can be checked in a similar way.

Case 4. $t \in [c_m, d_m] - [c_{m+1}, d_{m+1}]$ for some $m < n$. This case is similar to Case 2 (if $m < n - 1$) or to Case 3 (if $m = n - 1$) above, with the roles of f and g exchanged. ■

CLAIM 4. *f and g intersects at a unique point $\mathbf{z}_0$ and $\mathbf{z}_0$ is not computable in polynomial time.*

Proof. From property (iv) of function f_n, we know that $u_{n+1} = \frac{1}{4} + (2\phi(0^{n-1}) + 1)2^{-(p(n+1)+2)}$ is between $\frac{1}{4}$ and $\frac{3}{4}$ and is a dyadic rational whose binary expansion has length $\leq p(n + 1) + 2$ bits in the fractional part. Since $\alpha_{n+1} = \alpha_n + 2^{-q_2(n)}u_{n+1}$, it is clear that the first $q_2(n + 1)$ bits of the fractional part of α_0 is exactly α_{n+1}; or, equivalently, the substring of α_0 from the $(q_2(n) + 1)$th bit to the $q_2(n + 1)$th bit encodes exactly u_{n+1}. The claim then follows from the assumption that $\phi \notin PF_1$. ■

6. FINAL REMARKS

We have proved two lower bound results showing that if $PMLF_1 \not\subseteq PF_1$ then the fixed points and the intersection points of polynomial-time computable functions are not necessarily polynomial-time computable, even if the functions are well-conditioned. Many questions, however, still remain open. First, for the fixed points in the two-dimensional case, there is still a gap between the best known upper bound *PDLF* and the new lower bound $PMLF_1$. The precise complexity of the problem probably depends on the complexity of the discrete problem SPERNER$_2$, which appears to be between *PDLF* and $PMLF_1$. For the intersection point problem, we do not even know a better upper bound than the obvious bound *NP*. There appears much room for improvement.

Since the fixed point problem may be considered as a subproblem of the zero-finding problem, it is interesting to review the complexity of zeros of polynomial-time computable functions. In the two-dimensional case, Ko (1991) has found a lower bound UPF_1 for the zeros of a one-to-one, polynomial-time computable function on $[0, 1]^2$ that has a polynomial inverse modulus at zeros (i.e., is well-conditioned). This is a stronger lower bound; the function constructed there, however, is not a mapping from a square $[0, 1]^2$ to itself, although the unique zero $\mathbf{z}_0$ is indeed in $[0, 1]^2$. The following stronger lower bound on fixed points can be obtained by a simple modification of this result.

THEOREM 6.1 (Ko, 1991). *Assume that $P_1 \neq UP_1 \cap co\text{-}UP_1$. Then, there exists a polynomial-time computable function f mapping* $[0, 1]^2$ *to* $\mathbf{R}^2$ *such that* (i) *f has a unique fixed point $\mathbf{z}_0$,* (ii) *f is well-conditioned* (*with respect to the fixed-point problem*), *and* (iii) *$\mathbf{z}_0$ is not polynomial-time computable.*

A similar stronger lower bound exists for the zeros of one-dimensional, polynomial-time computable functions that have polynomial inverse moduli at zeros (but not necessarily one-to-one). We remark that the function f constructed above is not a mapping from a square $[a, b] \times [c, d]$ to itself, and the construction depends heavily on this property. The requirement that f maps a square to itself allows the application of Sperner's lemma and appears to have lowered the complexity of the fixed points.

Finally, we note that the relation between the new complexity classes *PDLF*, *PMLF*, and the classes *UP* and *PLS* (the class of problems solvable by polynomial local searches) is critical to our understanding of the complexity of fixed points and intersection points.

ACKNOWLEDGMENT

The author thanks Professor Arthur Chou for interesting discussions on fixed points.

REFERENCES

HIRSCH, M. D., PAPADIMITRIOU, C. H., AND VAVASIS, S. A. (1989), Exponential lower bounds for finding Brouwer fixed points, *J. Complexity* **5**, 379–416.

JOHNSON, D. S., PAPADIMITRIOU, C. H., AND YANNAKAKIS, M. (1988), How easy is local search? *J. Comput. System Sci.* **37**, 79–100.

KO, K. (1991), "Complexity Theory of Real Functions," Birkhäuser, Boston.

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

PAPADIMITRIOU, C. H. (1990), On graph-theoretic lemmata and complexity classes, *in* "Proceedings, 31st IEEE Symposium on Foundation of Computer Science, pp. 794–801.

POUR-EL, M., AND RICHARDS, I. (1989), "Computability in Analysis and Physics," Springer-Verlag, Berlin.

SHAMIR, A. (1990), IP = PSPACE, *in* "31st IEEE Symposium on Foundations of Computer Science," pp. 11–15.

SHASHKIN, YU. A. (1991), "Fixed Points," transl. from Russian by V. Minachin, Amer. Math. Soc., Providence, RI.