# Matching Algorithmic Bounds for Finding a Brouwer Fixed Point

XI CHEN

*Tsinghua University, Beijing, China*

AND

XIAOTIE DENG

*City University of Hong Kong, Hong Kong, China*

Abstract. We prove a new discrete fixed point theorem for direction-preserving functions defined on integer points, based on a novel characterization of boundary conditions for the existence of fixed points. The theorem allows us to derive an improved algorithm for finding such a fixed point. We also develop a new lower bound proof technique. Together, they allow us to derive an asymptotic matching bound for the problem of finding a fixed point in a hypercube of any constantly bounded finite dimension.

Exploring a linkage with the approximation version of the continuous fixed point problem, we obtain asymptotic matching bounds for the complexity of the approximate Brouwer fixed point problem in the continuous case for Lipschitz functions. It settles a fifteen-years-old open problem of Hirsch, Papadimitriou, and Vavasis by improving both the upper and lower bounds.

Our characterization for the existence of a fixed point is also applicable to functions defined on nonconvex domains, which makes it a potentially useful tool for the design and analysis of algorithms for fixed points in general domains.

Categories and Subject Descriptors: F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems—*Computations on discrete structures; sorting and searching; geometrical problems and computations*

---

An extended abstract of this article appeared in *Proceedings of the 37th Annual ACM Symposium on Theory of Computing* (*STOC*), 323–330.

X. Chen's work was supported by the Chinese National Key Foundation Plan (2003CB317807, 2004CB318108), the National Natural Science Foundation of China Grant 60553001 and the National Basic Research Program of China Grant (2007CB807900, 2007CB807901).

Part of the work of X. Chen was done while visiting City University of Hong Kong.

X. Deng's work was partially supported by a CERG grant of Research Grants Council of Hong Kong SAR (Project No. CityU 112707).

Authors' present addresses: X. Chen, School of Mathematics, Institute for Advanced Study, Princeton, NJ 08540; e-mail: csxichen@gmail.com; X. Deng, Department of Computer Science, City University of Hong Kong, Hong Kong, China, e-mail: csdeng@cityu.edu.hk.

General Terms: Algorithms, Economics, Theory

Additional Key Words and Phrases: Approximate fixed point, fixed point theorem, Lipschitz function, Sperner's lemma

**ACM Reference Format:**

Chen, X. and Deng, X. 2008. Matching algorithmic bounds for finding a Brouwer fixed point. J. ACM 55, 3, Article 13 (July 2008), 26 pages. DOI = 10.1145/1379759.1379761 http://doi.acm.org/10.1145/1379759.1379761

## 1. *Introduction*

Brouwer's fixed point theorem and its variations have had a profound influence in mathematical sciences and applications, including approximation theory [Meinardus 1963], dynamical systems [Robinson 1999], game theory [Nash 1950], and the most popularly known of all, the theory of general equilibrium in Economics [Arrow and Debreu 1954]. While fixed point theorems are applied to establish fundamental theories, fixed point algorithms are used to solve important application problems, especially in many recent works for network communication: TCP network calculus, by Altman et al. [2002]; network edge pricing, by Cole et al. [2003]; multicast pricing, by Mehta et al. [2003]; and TCP queue management, by Low [2003]. Fixed point theorems also have other important applications in computer science, for example, the spectral analysis for numerical computation, by Spielman and Teng [1996].

Brouwer's [1910] fixed point theorem can be stated succinctly as follows: any continuous function $\mathcal{F}$ mapping $D = [0, 1]^d$ to itself has a fixed point $x \in D$ such that $\mathcal{F}(x) = x$. In various levels of generalities, it can be extended to different relaxed requirements on the function $\mathcal{F}$ and domain $D$. A discrete and combinatorial characterization of Brouwer's theorem is Sperner's lemma. It ensures that a certain labeling rule on vertices of a simplicial decomposition of a simplex $S^d$ guarantees the existence of a subsimplex with all vertices differently labeled. Naturally, it has been influential on the design of combinatorial algorithms for the fixed point problem, started in the 60's with Scarf's seminal work [Scarf 1967], which finds an approximate fixed point by finding a completely labeled primitive set based on a structure lemma similar to, but not the same as, Sperner's Lemma. Kuhn replaced the primitive sets by simplices and simplicial partitions [Kuhn 1968]. The simplicial approach has since been adopted by most general purpose fixed point algorithms developed later, such as the restart algorithm of Merrill [1972] and the homotopy algorithm of Eaves [1972].

Although many combinatorial algorithms based on simplicial structures are in the worst case exponential in computation time, it was conjectured that the performance of some such algorithms might be better. Hirsch et al. [1989] played down such a hope by proving a general exponential lower bound. On the other hand, for contractive Lipschitz functions, that is, $\mathcal{F}$ such that $|\mathcal{F}(x) - \mathcal{F}(y)| \leq c \cdot |x - y|$ with $c < 1$, the problem can be solved efficiently, for example, by the iteration algorithm of Banach [1922], Newton's method (see the works of Ortega and Rheinbolt [1970], Kellogg et al. [1976], and Smale [1976]), and the interior ellipsoid algorithm of Huang et al. [1999].

In this article, we study fixed point algorithms for general Lipschitz functions. Therefore, the iterative approach for contractive functions does not apply here. Our study is motivated by a particularly interesting recent discrete version of the fixed point problem, introduced by Iimura [2003] and Iimura et al. [2005], stating that any direction-preserving function $\mathcal{F}$ (a discrete analogue to the continuous function) that maps $D = N^d = \{0, 1, 2, \dots, n-1\}^d$ to itself, has a fixed point.

Iimura's proof [2003] is nonconstructive by extending the discrete function $\mathcal{F}$ to a continuous function such that the latter has a fixed point if and only if $\mathcal{F}$ has one. It is, therefore, not suitable for developing algorithms for finding a solution. The algorithmic results for approximating fixed points by Hirsch et al. [1989] on the other hand, have a natural extension for the discrete version, leading to a lower bound of $\Omega(n^{d-2})$ and an upper bound of $O(n^d)$ for the discrete fixed point problem on the grid $N^d$. Noticeably, for $d = 2$, Hirsch et al. have a matching bound of $\Theta(n)$ for $N^2$. Closer examination of their upper bound for the 2-dimensional case would reveal a boundary condition for a fixed point to exist: the winding number of the boundary is nonzero. Our upper bound relies on the establishment of a similar boundary condition for higher dimensions. We exploit the grid structure and the direction-preserving condition of the discrete fixed point problem to develop a succinct combinatorial theorem that leads to the design of our algorithm.

In addition, our result gives an independent and constructive proof for Iimura's discrete fixed point theorem. In fact, we derive a general characterization for a pair of function and domain $(\mathcal{F}, D)$ to have a fixed point, which is also applicable to nonconvex domains, and thus improve the results of Iimura [2003] and Iimura et al. [2005].

The theorem is based on a characterization, via a parity argument, of $\mathcal{F}(x) - x$ on vertices of a unit cube, which is then applied onto a collection of unit cubes to establish a global boundary condition, if no fixed point exists. It is as elementary as Sperner's Lemma [Sperner 1928]. In fact, Brouwer's fixed point theorem for the continuous case can also be derived from our combinatorial theorem.

The tight lower bound proof for the two-dimensional case by Hirsch et al. is also very elegantly done. However, it is not suitable to extend to higher dimensions directly. Our lower bound proof fully utilizes the simplification brought in by the discrete version, and introduces a game on lattice graphs to focus on the essence of the problem. Then we extend the result derived from the game to obtain a matching lower bound for the discrete fixed point problem. Even though the final proof is deep and rather complicated, the approach is quite clear and accessible.

Finally, it is not hard to establish a linkage between direction-preserving functions for the discrete fixed point problem and Lipschitz functions for the approximate fixed point problem. That linkage makes our results for the discrete version extendable to the approximate fixed point problem for Lipschitz functions. In particular, our results solve an open problem proposed by Hirsch et al. [1989].

For succinctness of the presentation here, we sometimes, especially in Section 3 and 4, consider the function $f(x) = \mathcal{F}(x) - x$. The problem of finding a fixed point of $\mathcal{F}$ is equivalent to the problem of finding a root of $f$: $\mathcal{F}(x) = x \Leftrightarrow f(x) = 0$. We call such a point a zero point of $f$. As pointed out by Hirsch et al. [1989], the general zero point problem is algorithmically harder than the fixed point problem. However, our discussion here only considers a special version of the zero point problem which is equivalent to the discrete fixed point problem.

The article is organized as follows. In Section 2, we give the necessary definitions, together with a proof that the above two problems are computationally equivalent. The crucial combinatorial lemma is then discussed in Section 3, followed by the algorithm and the upper bound proof. The lower bound construction and proof are presented in Section 4. In Section 5, we discuss applications to the continuous case. We conclude in Section 6 with discussions and remarks on our approach and other related results as well as potential research directions.

## 2. *Definitions*

We start with some notation. For any nonzero $x \in \mathbb{R}$, we let $\text{sgn}(x) = 1$ if $x > 0$, and $\text{sgn}(x) = -1$ if $x < 0$. For any $1 \le k \le d$, we use $e^k$ to denote the $k$th unit vector of $\mathbb{Z}^d$, where $e^k_k = 1$ and $e^k_i = 0$ for all $1 \le i \ne k \le d$. For any vector $v \in \mathbb{Z}^d$, $1 \le k \le d$, and $l \in \mathbb{Z}$, we define vector $v[k \leftarrow l] = v + (l - v_k)e^k$. For simplicity, we use $v^-$ to denote $v[d \leftarrow (v_d - 1)]$ and $v^+$ to denote $v[d \leftarrow (v_d + 1)]$.

*Definition* 2.1. For any $p < q \in \mathbb{Z}^d$ (i.e., $p_i < q_i$ for all $1 \le i \le d$), we define a rectangular set $A_{p,q} = \{r \in \mathbb{Z}^d \mid p \le r \le q\}$. Its boundary is then defined as

$$B_{p,q} = \{r \in A_{p,q} \mid \exists\, 1 \le i \le d \text{ such that } r_i = p_i \text{ or } q_i\}.$$

*Definition* 2.2. Map $\mathcal{F} : A_{p,q} \to \mathbb{R}^d$ is said to be direction-preserving if for any $r^1, r^2 \in A_{p,q}$ such that $|r^1 - r^2|_\infty \le 1$, we have $(\mathcal{F}_i(r^1) - r^1_i)(\mathcal{F}_i(r^2) - r^2_i) \ge 0$, for all $1 \le i \le d$.

*Definition* 2.3. Function $f : S \to \{0, \pm e^1, \pm e^2, \ldots, \pm e^d\}$, where $S \subset \mathbb{Z}^d$, is said to be direction-preserving if for any $r^1, r^2 \in S$ such that $|r^1 - r^2|_\infty \le 1$, we have $|f(r^1) - f(r^2)|_\infty \le 1$. We let $F[S]$ denote the set of all such functions on $S$.

Function $f : A_{p,q} \to \{0, \pm e^1, \ldots, \pm e^d\}$ is said to be bounded if $\mathcal{F}(r) = f(r) + r$ is a map from $A_{p,q}$ to itself. Using the results of Iimura [2003] and Iimura et al. [2005], we have the following theorem. Later, we will derive for it a new constructive proof in Section 3.

THEOREM 2.4. *For any direction-preserving map $\mathcal{F}$ from $A_{p,q}$ to itself, there exists $r^* \in A_{p,q}$ such that $\mathcal{F}(r^*) = r^*$. Such a point $r^*$ is called a fixed point of $\mathcal{F}$.*

We are interested in the algorithmic complexity of the following discrete fixed point problem $\mathbf{DFP}^d$ : given a direction-preserving map $\mathcal{F}$ from $A_{p,q}$ to itself with $A_{p,q} \subset \mathbb{Z}^d$, find a fixed point $r^*$ of $\mathcal{F}$. Algorithms discussed in this paper will be restricted to those that are based on map evaluations. That is, map $\mathcal{F}$ is provided as an oracle to algorithm designers. It can only be accessed by calling the oracle to evaluate $\mathcal{F}(r)$ when a point $r \in A_{p,q}$ is given. We use $n = |p - q|_\infty + 1$ as the measure of input size, then two kinds of complexities are considered here: query complexity $Q(n, d)$ and time complexity $T(n, d)$.

We will focus on the study of the following discrete zero point problem $\mathbf{DZP}^d$: given a bounded function $f \in F[A_{p,q}]$, where $A_{p,q} \subset \mathbb{Z}^d$, compute a zero point $r^*$ such that $f(r^*) = 0$. The existence of such an $r^*$ is guaranteed by Theorem 2.4, as (**A**) $\mathcal{F}(r) = f(r) + r$ is a direction-preserving map from $A_{p,q}$ to itself. Similarly, we use $Q'(n, d)$ and $T'(n, d)$ to denote the query and time complexity of $\mathbf{DZP}^d$, respectively. Our main results in Section 3 and 4 are

THEOREM 2.5. *For any $d \geq 2$ and $n > 48d$, we have*

$$0.5(\lfloor (n-1)/2^{10} \rfloor)^{d-1} \leq Q'(n,d) \leq 7n^{d-1} \quad \text{and}$$
$$Q'(n,d) \leq T'(n,d) \leq O(d^2(2n)^{d-1}).$$

There is a strong relationship between these two problems. Statement (**A**) shows that any algorithm for $\mathbf{DFP}^d$ can be used to solve $\mathbf{DZP}^d$: We simply execute it on $\mathcal{F}$ to find a fixed point $r^*$ of $\mathcal{F}$ and it must also be a zero point of $f$. Whenever the algorithm wants to evaluate $\mathcal{F}(r)$, we query $f(r)$ and compute $\mathcal{F}(r) = f(r) + r$ in $O(d)$ time. This reduction shows that

$$Q'(n,d) \leq Q(n,d) \quad \text{and} \quad T'(n,d) \leq O(d) \cdot Q(n,d) + T(n,d).$$

On the other hand, given any direction-preserving map $\mathcal{F}$ from $A_{p,q}$ to itself, a bounded function $f \in F[A_{p,q}]$ can be constructed as follows: For every $r \in A_{p,q}$, if $\mathcal{F}(r) = r$, then set $f(r) = 0$; otherwise, set $f(r) = \text{sgn}(\mathcal{F}_i(r) - r_i)e^i$, where $i$ is the largest index such that $\mathcal{F}_i(r) \neq r_i$. Similarly, we have

$$Q(n,d) \leq Q'(n,d) \quad \text{and} \quad T(n,d) \leq O(d) \cdot Q'(n,d) + T'(n,d).$$

In conclusion, $\mathbf{DFP}^d$ and $\mathbf{DZP}^d$ are equivalent in computational complexity and Theorem 2.5 also holds for $Q(n,d)$ and $T(n,d)$. For any constant $d \geq 2$, it gives a matching algorithmic bound of $\Theta(n^{d-1})$ for both complexities.

## 3. *An Algorithm for the Discrete Zero Point Problem*

In this section, we first prove a discrete zero point theorem. For any $f \in F[A_{p,q}]$, it gives us a condition on $f(B_{p,q})$, which guarantees the existence of a zero point in $A_{p,q}$. Then, a divide-and-conquer algorithm for $\mathbf{DZP}^d$ is presented: Recursively, we divide the function domain of $f$ into two parts (of almost the same size); the discrete zero point theorem is then applied to decide which side to follow.

3.1. THE DISCRETE ZERO POINT THEOREM. First, we define subsets of $\mathbb{Z}^d$ called $t$-cubes where $0 \leq t \leq d$. Lemma 3.2 and 3.3 about $t$-cubes are both easy to prove.

*Definition* 3.1. For any $r \in \mathbb{Z}^d$ and $S \subset \{1, 2, \ldots, d\}$ with $|S| = d - t$, the $t$-cube $C^t \subset \mathbb{Z}^d$ which is centered at $r$ and perpendicular to $S$ is defined as

$$C^t = \{p \in \mathbb{Z}^d \mid \forall\, 1 \leq i \leq d, \text{ if } i \in S, \text{ then } p_i = r_i; \text{ otherwise, } p_i = r_i \text{ or } r_i + 1\}.$$

For any rectangular set $A_{p,q} \subset \mathbb{Z}^d$, we use $V[p,q]$ to denote the set of all $(d-1)$-cubes $C \subset B_{p,q}$ (every such $C$ is perpendicular to $S = \{k\}$ for some $k : 1 \leq k \leq d$ and centered at $r$ with $r_k = p_k$ or $q_k$).

LEMMA 3.2. *Let $C^t$ be a $t$-cube in $\mathbb{Z}^d$, where $t \geq 2$. Then for every $(t-2)$-cube $C^{t-2} \subset C^t$, there are exactly two $(t-1)$-cubes in $C^t$ that contain $C^{t-2}$.*

LEMMA 3.3. *Let $C^{d-1}$ be a $(d-1)$-cube in $A_{p,q} \subset \mathbb{Z}^d$. If $C^{d-1} \in V[p,q]$, then it is contained by exactly one $d$-cube in $A_{p,q}$; otherwise, it is contained by two $d$-cubes.*

Inductively, we define bad $t$-cubes $C^t \subset \mathbb{Z}^d$ with respect to a function $f$ from $C^t$ to $\{0, \pm e^1, \ldots, \pm e^d\}$, where $0 \leq t \leq d-1$, as follows.

*Definition* 3.4. A 0-cube $\{r\} = C^0 \subset \mathbb{Z}^d$ is bad relative to $f$ if $f(r) = e^1$.
For $1 \le t \le d-1$, a $t$-cube $C^t \subset \mathbb{Z}^d$ is bad relative to $f$ if

**(B$_1$).** $f(C^t) = \{e^1, e^2, \ldots, e^{t+1}\}$ (where $f(C^t) = \{f(r), r \in C^t\}$); and
**(B$_2$).** the number of bad $(t-1)$-cubes in $C^t$ is odd.

For any $f \in F[A_{p,q}]$, we let $V_f[p,q]$ denote the set of all bad $(d-1)$-cubes in $V[p,q]$.

LEMMA 3.5. *For any $d$-cube $C^d \subset \mathbb{Z}^d$ and $f \in F[C^d]$, such that $f$ has no zero point in $C^d$, the number of bad $(d-1)$-cubes in $C^d$ must be even.*

Actually, Lemma 3.5 is a direct corollary of Lemma 3.7. Before presenting the proof, we note that Lemma 3.5 together with Lemma 3.3 imply the following zero point theorem.

THEOREM 3.6. *If $|V_f[p,q]|$ is odd, then $f \in F[A_{p,q}]$ must have a zero point.*

LEMMA 3.7. *For any $t$-cube $C^t \subset \mathbb{Z}^d$, where $1 \le t \le d$, and any $f \in F[C^t]$ such that $f(C^t) \subset \{\pm e^1, \pm e^2, \ldots, \pm e^t\}$, the number of bad $(t-1)$-cubes in $C^t$ must be even.*

PROOF. We use mathematical induction on $t$. The base case for $t = 1$ is trivial. For the case when $t \ge 2$, we assume that the claim is true for $t-1$. First, if there is no bad $(t-1)$-cube in $C^t$, then we are done. Otherwise, there exists at least one bad $(t-1)$-cube $C^{t-1} \subset C^t$. Condition (**B$_1$**) shows that $f(C^{t-1}) = \{e^1, e^2, \ldots, e^t\}$, and the direction-preserving property of $f$ requires that $f(C^t) = \{e^1, e^2, \ldots, e^t\}$.

Now for any $(t-1)$-cube $C^{t-1} \subset C^t$, we prove that if it satisfies condition (**B$_2$**), then it must also satisfy (**B$_1$**). This shows that $C^{t-1}$ is bad iff the number of bad $(t-2)$-cubes in it is odd. As a result, the parity of the number of bad $(t-1)$-cubes in $C^t$ is the same as the parity of $\sum_{C^{t-1} \subset C^t} |\{\text{bad } (t-2)\text{-cubes in } C^{t-1}\}|$. Lemma 3.7 then follows directly from the fact that the latter summation is even (due to Lemma 3.2).

Therefore, to finish the proof, we only need to prove that (**B$_2$**) implies (**B$_1$**) for any $C^{t-1} \subset C^t$. Suppose $C^{t-1} \subset C^t$ satisfies (**B$_2$**). Since there is at least one bad $(t-2)$-cube in $C^{t-1}$, we have $\{e^1, \ldots, e^{t-1}\} \subset f(C^{t-1})$. If $f(C^{t-1}) = \{e^1, \ldots, e^{t-1}\}$, then by the inductive hypothesis, the number of bad $(t-2)$-cubes in $C^{t-1}$ must be even, which contradicts property (**B$_2$**). Therefore, $f(C^{t-1}) \subset f(C^t) = \{e^1, \ldots, e^t\}$ must equal to $\{e^1, \ldots, e^t\}$ and property (**B$_1$**) is satisfied. □

## 3.2. THE RECURSIVE ALGORITHM.

We are now ready to present the recursive algorithm called **FindZero**$^d(g, V_g[p,q])$. Its input satisfies $g \in F[A_{p,q}]$, where $A_{p,q} \subset \mathbb{Z}^d$, and $|V_g[p,q]|$ is odd. Its output is a zero point of $g$.

If $|p-q|_\infty = 1$, then the algorithm simply queries all the vertices and returns a zero point (whose existence is guaranteed by Theorem 3.6). Otherwise, it divides $A_{p,q}$ into two parts and proceeds recursively on the bad part. Let $n = |p-q|_\infty + 1$. The algorithm uses at most $6n^{d-1}$ queries and $O(d^2(2n)^{d-1})$ time to find a zero point of $g$, for all $d \ge 2$ and $n > 48d$.

But how can we use **FindZero**$^d$ to solve **DZP**$^d$? Given a bounded $f \in F[A_{p,q}]$, we first extend $f$ to be $f'$ on $A_{p',q'}$, where $p' = p-1$ and $q' = q+1$, as follows. First, $f' = f$ on $A_{p,q}$. Then for every $r \in B_{p',q'}$, let $i$ be the largest integer such

```
Algorithm  Cut^d (g, V_g[p, q], k)

Ensure: $1 \le k \le d$, $q_k - p_k > 1$ and $|V_g[p,q]|$ is odd
1: set $l = \lfloor (p_k + q_k)/2 \rfloor$, $p' = p[k \leftarrow l]$, $q' = q[k \leftarrow l]$ and $V_g[p,q'] = V_g[p',q] = \emptyset$
2: for any $r \in S = \{r \in \mathbb{Z}^d \mid \forall\, 1 \le i \le d,\ p'_i \le r_i \le q'_i\}$, query $g(r)$
3: for any $(d-1)$-cube $C \in V_g[p,q]$ do
4:     if $C \in V[p,q']$, then $V_g[p,q'] = V_g[p,q'] \cup \{C\}$, else $V_g[p',q] = V_g[p',q] \cup \{C\}$
5: compute $V = \{$bad $(d-1)$-cubes in $S\}$
6: for any $(d-1)$-cube $C \in V$, add $C$ into both $V_g[p,q']$ and $V_g[p',q]$
7: if $|V_g[p,q']|$ is odd, then output $(g, V_g[p,q'])$, else output $(g, V_g[p',q])$
```

```
Algorithm  FindZero^d(g, V_g[p, q])

Ensure: $p < q$ and $|V_g[p,q]|$ is odd
1: if $|p-q|_\infty = 1$, then query every point $r \in A_{p,q}$ and output a zero point of $g$
2: for $i = 1$ to $d$ do
3:     if $q_i - p_i > 1$, then set $(g, V_g[p,q]) = \mathbf{Cut}^d(g, V_g[p,q], i)$
4: output $\mathbf{FindZero}^d(g, V_g[p,q])$
```

FIG. 1. Details of algorithm **FindZero**$^d$.

that $r_i = p'_i$ or $q'_i$. If $r_i = p'_i$, then $f'(r) = +e^i$; otherwise, $f'(r) = -e^i$. It is easy to check that $f' \in F[A_{p',q'}]$. Lemma 3.8 below is proved in Appendix A.

LEMMA 3.8. *For any bounded function $f \in F[A_{p,q}]$, $V_{f'}[p',q']$ contains exactly one $(d-1)$-cube, that is, the one that is centered at $p'$ and perpendicular to set $\{1\}$.*

As a result, we can call **FindZero**$^d(f', V_{f'}[p',q'])$ to find a zero point of $f$. This gives us the following upper bounds:

$$Q'(n,d) \le 6(n+2)^{d-1} \le 7n^{d-1} \quad \text{and} \quad T'(n,d) = O(d^2(2n)^{d-1}),$$

for any $d \ge 2$ and $n > 48d$.

The algorithm is described in Figure 1 above. Here **Cut**$^d$ uses $S$, which is perpendicular to the $k$th dimension, to divide $A_{p,q}$ into two smaller sets of almost the same size. After querying all the points in $S$, it chooses one set that still satisfies the condition of our zero point theorem to return.

Finally, we analyze the complexity of **FindZero**$^d$. Let $n = |p-q|_\infty + 1$. Then the number of queries used by the $d$ calls to **Cut**$^d$ in **FindZero**$^d$ is at most

$$n^{d-1} + n^{d-2}(\lfloor n/2 \rfloor + 1) + \cdots + (\lfloor n/2 \rfloor + 1)^{d-1} < 3n^{d-1},$$

under the condition that $n > 48d$, and recurrence can be solved to derive a $6n^{d-1}$ upper bound for the query complexity of **FindZero**$^d$. An implementation of line 5 in **Cut**$^d$, based on dynamic programming, can be found in Appendix B. It uses $O(d^2 2^d |S|)$ time, so the $d$ calls to **Cut**$^d$ in **FindZero**$^d$ use $O(d^2 2^d n^{d-1})$ time. This gives a $O(d^2(2n)^{d-1})$ upper bound for the time complexity of **FindZero**$^d$.

## 3.3. A NEW DISCRETE FIXED POINT THEOREM.

Now we see that Theorem 2.4 follows directly from Theorem 3.6 and Lemma 3.8. Actually, Theorem 3.6 implies a stronger fixed point theorem. Given any direction-preserving map $\mathcal{F}$ from $A_{p,q}$ to $\mathbb{R}^d$, one can construct a function $f \in F[A_{p,q}]$ using the method in Section 2, then

COROLLARY 3.9. *If $|V_f[p,q]|$ is odd, then $\mathcal{F}$ must have a fixed point.*

Furthermore, the way we prove Theorem 3.6 suggests that both Theorem 3.6 and Corollary 3.9 can be easily generalized to nonconvex domain $D \subset \mathbb{Z}^d$ which is a union of $d$-cubes.

## 4. *A Lower Bound for the Discrete Zero Point Problem*

In this section, we first define a class of undirected graphs $G_{m,d} = (N_{m,d}, E_{m,d})$. Then a game on $G_{m,d}$, of a hidden-seek type, between two players, Alex and Bob, is introduced. We analyze the minimum number of queries needed by Bob and finally, use the lower bound for Bob to derive a lower bound for the query complexity of problem $\mathbf{DZP}^d$.

### 4.1. DEFINITIONS OF PIPE PATHS AND SPARSE SETS IN GRAPH $G_{m,d}$.

We start with the definition of graph $G_{m,d}$. For any $m \geq 2$, we define

$$N_{m,d} = \{r \in \mathbb{Z}^d \mid \forall\, 1 \leq i \leq d,\ 1 \leq r_i \leq m\}.$$

For $S \subset \mathbb{Z}^d$ and $t \in \mathbb{Z}$, the layer $t$ of set $S$ is defined as $S^t = \{r \in S, r_d = t\}$.

*Definition* 4.1. For any $d \geq 1$ and $m \in \mathbb{Z}^+$ that is a multiple of 256, we define graph $G_{m,d} = (N_{m,d}, E_{m,d})$ as follows. For all vertices $u, v \in N_{m,d}$, $uv \in E_{m,d}$ iff there exists $1 \leq i \leq d$ such that $|u_i - v_i| = 1$ and $u_j = v_j$ for all other $1 \leq j \leq d$.

For each $t : 1 \leq t \leq m$, the layer $t$ of graph $G_{m,d}$ is the subgraph spanned by $N^t_{m,d}$. Obviously, it is isomorphic to $G_{m,d-1}$ under the mapping $D$ where $D(u) = (u_1, u_2, \ldots, u_{d-1})$. Given a path $P = u \ldots w$ in $G_{m,d}$, $u$ is called the start vertex and $w$ is called the end vertex of $P$. If the end vertex of path $P_1$ is same as the start vertex of path $P_2$, then we use $P_1 \cup P_2$ to denote the concatenation of $P_1$ and $P_2$.

*Definition* 4.2. Path $P = v^1 v^2 \ldots v^k$ in $G_{m,d}$ is said to be monotone if $k = 1$ or

$$v^1_d + 1 = v^2_d \leq \cdots \leq v^{k-1}_d = v^k_d - 1 \quad \text{or} \quad v^1_d - 1 = v^2_d \geq \cdots \geq v^{k-1}_d = v^k_d + 1.$$

For any $v^1_d \leq t \leq v^k_d$, we use $P^t$ to denote the part of $P$ on layer $t$ of $G_{m,d}$. It is clear that for any monotone path $P$, $P^t$ is a sub-path of $P$ on layer $t$ of $G_{m,d}$.

Next we define pipe paths and sparse sets in graph $G_{m,d}$. Proofs of Lemma 4.5 and 4.6 are presented in Appendix C and D, respectively.

*Definition* 4.3. For $d = 1$, every path $P$ in $G_{m,1}$ is a pipe path.
For $d \geq 2$, $P$ is a pipe path in $G_{m,d}$ if it is monotone and for any $1 \leq t \leq m$, $P^t$ is either empty or a pipe path in the layer $t$ of $G_{m,d}$ (in another word, $D(P^t)$ is a pipe path in graph $G_{m,d-1}$).

*Definition* 4.4. Let $S$ be a subset of $N_{m,d}$ and $u$ be a vertex of $G_{m,d}$. For $d = 1$, $S$ is said to be sparse relative to $u$ (in $G_{m,1}$) if $S = \emptyset$. For $d \geq 2$, $S$ is said to be sparse relative to $u$ if it satisfies the following three conditions:

(1) $u \notin S$ and $|S| < l_{m,d} = (m/256)^{d-1}$;
(2) If $u_d < m$, then $S^{u_d+1}$ is sparse relative to $u^+$ in layer $u_d + 1$ of $G_{m,d}$;
(3) If $u_d > 1$, then $S^{u_d-1}$ is sparse relative to $u^-$ in layer $u_d - 1$ of $G_{m,d}$.

We let $K_{m,d}$ denote the set of all pairs $(S, u)$ such that $S$ is sparse relative to $u$.

Clearly, $S = \emptyset$ is sparse with respect to $u$ at any dimension. Condition (2) (and (3) similarly) in the definition means that, set $D(S^{u_d+1})$ (or $D(S^{u_d-1})$) is sparse relative to $D(u^+)$ (or $D(u^-)$) in graph $G_{m,d-1}$.

LEMMA 4.5. *For any $S \subset G_{m,d}$,*

$$|\{u \in G_{m,d} \text{ such that } (S, u) \notin K_{m,d}\}| \leq 256^{d-1} m |S|.$$

LEMMA 4.6. *For any pair $(S, u) \in K_{m,d}$, where $m \geq 12d$, there exists a set of $m^d/2$ pipe paths $\{P_1, P_2, \ldots, P_{m^d/2}\}$ such that*

(1) *path $P_i$ starts at $u$ and $P_i \cap S = \emptyset$, for all $i : 1 \leq i \leq m^d/2$; and*
(2) *the $m^d/2$ end vertices of the $m^d/2$ paths are distinct.*

## 4.2. PIPE PATH FINDING ON GRAPH $G_{m,d}$.

We define a pipe path finding problem on $G_{m,d}$. We present it as a game between two players: Alex and Bob. At the beginning of the game, Bob picks a pair $(S, u)$ from $K_{m,d}$ and shows it to Alex. Alex then picks a pipe path $P$ in graph $G_{m,d}$, starting at $u$ and satisfying $P \cap S = \emptyset$. Bob's goal is to find it out by a sequence of queries. We will prove a lower bound on the number of queries needed by Bob.

At each round, Bob sends a vertex $v \in G_{m,d}$ to Alex. Alex is an oblivious player and answers the query of Bob according to his pipe path $P$:

*Case* 1. If $v$ is not on $P$, Alex returns "false";
*Case* 2. If $v$ is on $P$, but it is not the ending vertex of $P$, then Alex returns the sub-path $R$ of $P$ which starts at $u$, passes $v$, and ends at the successor of $v$;
*Case* 3. If $v$ is the ending vertex of $P$, Alex returns "true" and the whole path $P$; The pipe path is found and Bob wins.

We allow Bob to make a total of $l_{m,d} = (m/256)^{d-1}$ rounds of queries. After each query, Bob knows more information about the pipe path $P$ held by Alex. When Bob uses up all the $l_{m,d}$ queries, Alex must reveal the pipe path $P$ to convince Bob that he has followed the rules honestly.

A (deterministic) strategy of Bob includes: (1) an initial pair $(S, u) \in K_{m,d}$ that starts the game; and (2) how to choose a query vertex $v \in G_{m,d}$, given the query-answer history so far.

We will prove the following theorem:

THEOREM 4.7. *If $m \geq 12d$, then for every strategy of Bob, there is at least one pipe path $P$ in $G_{m,d}$, which requires more than $l_{m,d}$ rounds of queries.*

To prove this lower bound, we introduce a malicious player: Alice, in the place of Alex. Different from Alex, Alice does not choose a pipe path at the beginning of the game. Instead, she will use a constructive way to derive a pipe path in $G_{m,d}$ (see Theorem 4.8 for details) that beats Bob's effort to win the game.

For every graph $G_{m,d}$ such that $m \geq 12d$, we construct a strategy $T[m, d]$ for Alice so that she can always win the game. The strategy $T[m, d]$ is composed of three modules: **Init**$(S, u)$, **Query**$(v)$ and **GetPath**(). Alice can use it to play the game against (any strategy of) Bob in the following way:

(1) At the initiation stage of receiving the pair $(S, u) \in K_{m,d}$, Alice initiates the strategy $T[m, d]$ by calling **Init**$(S, u)$;
(2) At each round, when vertex $v \in G_{m,d}$ is queried by Bob, Alice calls **Query**$(v)$ and answers Bob with its output; (The output of **Query**$(v)$ is either "false" or a path that starts at $u$, passes $u$, and ends at the successor of $u$.)
(3) Finally, after answering all the $l_{m,d}$ queries, Alice calls **GetPaths**() (with no input), which outputs a collection of pipe paths in $G_{m,d}$. Every path in it is consistent with the query-answer history (for details, see the notations below). As a result, Alice can reveal any of them to Bob, and wins the game.

We need the following notations: After the first $s \leq l_{m,d}$ queries, we let $U_s$ denote the set of all paths answered by Alice so far, and

$$H_s = S \cup \{v,\ v \text{ is queried by Bob in the first } s \text{ rounds and Alice's answer is "false"}\}.$$

Set $H_s$ is also called the forbidden set before the $(s+1)^{st}$ round.

After the first $s \leq l_{m,d}$ queries, a pipe path $P$ in $G_{m,d}$ (starting with $u$) is said to be consistent with the query-answer history $(U_s, H_s)$, if all the paths in $U_s$ are prefixes of $P$, and $P \cap H_s = \emptyset$. Both the construction of $T[m, d]$ and the proof of the following theorem are presented in Appendix E.

THEOREM 4.8. *If $m \geq 12d$, then for any strategy of Bob, $T[m, d]$ satisfies:*

**C**$_1$ *After being initiated by a pair $(S, u) \in K_{m,d}$, the output of* **Query**$(v)$ *is either "false", or a path starts at $u$, passes $v$, and ends at the successor of $v$;*
**C**$_2$ *After all the $l_{m,d}$ queries,* **GetPaths**() *outputs a collection of at least $(m^d/8)$ pipe paths in $G_{m,d}$. They all start at $u$ and their ending vertices are distinct. Each of them is consistent with the query-answer history.*

Theorem 4.8 shows that, no matter what strategy Bob employs, there exists a pipe path $P$ (and many, as stated in the theorem) such that, if Alex picks $P$ at the beginning of the game, then he is able to answer all the $l_{m,d}$ queries correctly without revealing $P$ to Bob. Theorem 4.7 then follows from Theorem 4.8.

## 4.3. A LOWER BOUND FOR THE DISCRETE ZERO POINT PROBLEM.

We now apply Theorem 4.7 to derive a lower bound for the query complexity of **DZP**$^d$. The idea is to convert any algorithm for problem **DZP**$^d$ to a strategy for Bob in the path finding game.

To this end, we build, for every pipe path $P$ in $G_{m,d}$, a bounded and direction-preserving function $f_P \in F[N_{n,d}]$, where $n = 4m + 1$. The construction of $f_P$ is described in Appendix F, which is straight forward but tedious. Together with the construction, we also have a map $g$ which is independent of $P$ and maps every point $r \in N_{n,d}$ to a pair of vertices $g(r) = (v_1, v_2)$ in $G_{m,d}$. The construction has the following two nice properties:

**D**$_1$ For every pipe path $P$ in $G_{m,d}$, function $f_P$ has exactly one zero point $r^*$. Let $(v_1, v_2) = g(r^*)$, then $v_1 = v_2$ is the ending vertex of $P$;

**D**$_2$ For any point $r \in N_{n,d}$, to decide $f_P(r)$, one only need to know the relationship between the pipe path $P$ and $v_1, v_2$, where $(v_1, v_2) = g(r)$. In particular, if $P$ is the secret path held by Alex, Bob can decide $f_P(r)$ by querying $v_1$ and $v_2$.

Now given an algorithm for $\mathbf{DZP}^d$, we can transform it into a strategy of Bob in the path finding game on $G_{m,d}$ as follows (let $P$ denote the path held by Alex):

(1) At the beginning of the game, Bob sends $((1, 1, \ldots, 1), \emptyset)$ to Alex. Then Bob starts to run the algorithm for $\mathbf{DZP}^d$, and asks it to find a zero point of a bounded function in $F[N_{n,d}]$, where $n = 4m + 1$;

(2) Whenever the algorithm for $\mathbf{DZP}^d$ needs to evaluate the function at point $r \in N_{n,d}$. Bob queries Alex $v_1$ and $v_2$, where $g(r) = (v_1, v_2)$. If one of the answers is ("true", $P$), then Bob wins and the game is over (the strategy terminates). Otherwise, by Property **D**$_2$, Bob decides $f_P(r)$, and sends it to the algorithm for $\mathbf{DZP}^d$. By Property **D**$_1$, $f_P(r) \neq 0$.

Finally, we start a query-answer game on $G_{m,d}$, where $m \geq 12d$, in which Bob plays the strategy described above. By Theorem 4.7, there is a pipe path $P^*$ which Bob cannot find with $l_{m,d}$ queries. It is easy to check that, as the game proceeds, the algorithm for $\mathbf{DZP}^d$ evaluated $f_{P^*}$ for $\lceil l_{m,d}/2 \rceil$ times, but has not found any zero point yet. As a result of this reduction, we have

$$Q'(n,d) > \lceil l_{m,d}/2 \rceil \quad \text{and} \quad Q'(n,d) \geq 0.5(\lfloor (n-1)/2^{10} \rfloor)^{d-1},$$

for all $d \geq 2$ and $n > 48d$, which is exactly the lower bound for problem $\mathbf{DZP}^d$ in Theorem 2.5.

## 5. *Application to the Approximate Fixed Point Problem*

In this section, we first define the approximate fixed point problem $\mathbf{AFP}^{M,d,m}$ with respect to Lipschitz functions [Hirsch et al. 1989]. Then, we apply Theorem 2.5 to derive three bounds for its complexity.

*Definition* 5.1. Map $\mathcal{F} : E^d = [0,1]^d \to \mathbb{R}^d$ satisfies a Lipschitz condition with constant $L$ if for any $x, y \in E^d$, $|\mathcal{F}(x) - \mathcal{F}(y)|_\infty \leq L|x - y|_\infty$.

We use $L_{M,d}$ to denote the set of all maps $\mathcal{F} : E^d \to E^d$ such that $\mathcal{F}(x) - x$ satisfies a Lipschitz condition with constant $M$.

By Brouwer's fixed point theorem, every $\mathcal{F} \in L_{M,d}$ has at least one fixed point $x^* \in E^d$ such that $\mathcal{F}(x^*) = x^*$. The approximate fixed point problem $\mathbf{AFP}^{M,d,m}$ is defined as follows: given a map $\mathcal{F} \in L_{M,d}$, find an approximate fixed point with error $2^{-m}$, that is, a point $x^* \in E^d$ such that $|\mathcal{F}(x^*) - x^*|_\infty \leq 2^{-m}$. Similarly, $\mathcal{F}$ is provided as an oracle to algorithm designers. It can only be accessed by calling the oracle to evaluate $\mathcal{F}(x)$, when a point $x \in E^n$ is given. We let $Q(M, d, m)$ and $T(M, d, m)$ denote the query complexity and time complexity of $\mathbf{AFP}^{M,d,m}$, respectively.

For the case when $d = 2$, Hirsch et al. [1989] shows that $Q(M, 2, m) = \Theta(2^m M)$. However, when $d > 2$, there is still a gap between their lower bound and upper bound. The main results of this section are

THEOREM 5.2. *For any $d$ and $m$ such that $d \geq 2$, $2^m M > 192d^3$ and $2^m > 4d$,*

$$0.25\left(\lfloor n_2/2^{11}\rfloor\right)^{d-1} - 1 \leq Q(M,d,m) \leq 8n_1^{d-1} \text{ and}$$
$$T(M,d,m) = O(d^2(2^{m+1}M)^{d-1}),$$

*where $n_1 = \lceil 2^m M\rceil$ and $n_2 = \lfloor 2^{m-2}M/d^2\rfloor$.*

For any specific constant $d$, it gives a matching bound of $\Theta((2^m M)^{d-1})$ for both complexities, thus settles an open problem in Hirsch et al. [1989].

5.1. TWO UPPER BOUNDS FOR THE APPROXIMATE FIXED POINT PROBLEM. Let $\mathcal{F} \in L_{M,d}$ be the input map. We first build a function $f \in F[A_{p,q}]$, where $p_i = 0$ and $q_i = n_1$ for all $1 \leq i \leq d$, as follows. For any $r \in A_{p,q}$, $f(r) \in \{0, \pm e^1, \ldots, \pm e^d\}$ is completely determined by $\mathcal{F}(x)$, where $x = r/n_1$. If $|\mathcal{F}(x) - x|_\infty \leq 2^{-m}$, then $f(r) = 0$. Otherwise, let $i$ be the largest index that satisfies $|\mathcal{F}_i(x) - x_i| > 2^{-m}$ and set $f(r) = \text{sgn}(\mathcal{F}_i(x) - x_i)e^i$. It is not hard to check that the Lipschitz property of $\mathcal{F}$ guarantees that $f$ is both bounded and direction-preserving.

Since $f$ is both bounded and direction-preserving, we can use any algorithm for problem $\mathbf{DZP}^d$ to find a zero point $r^*$ of $f$, and the construction of $f$ ensures that $x^* = r^*/n_1$ is an approximate fixed point of $\mathcal{F}$. Each time the algorithm queries $f(r)$, for some $r \in A_{p,q}$, we evaluate $\mathcal{F}$ at $x = r/n_1$ and use $O(d)$ time to compute $f(r)$. Therefore, we have

$$Q(M,d,m) \leq Q'(n_1+1,d) \quad \text{and}$$
$$T(M,d,m) \leq Q'(n_1+1,d)\cdot O(d) + T'(n_1+1,d) + O(d).$$

The two upper bounds in Theorem 5.2 then follows from Theorem 2.5.

5.2. A LOWER BOUND FOR THE APPROXIMATE FIXED POINT PROBLEM. Let $c = M/(2d)$ and $l = \lfloor c\rfloor$. Let $p$ and $q$ be two vectors in $\mathbb{Z}^d$ such that $p_i = 0$ and $q_i = n_2 + 2l$ for all $1 \leq i \leq d$. For every bounded function $f \in F[N_{n_2+1,d}]$, we build a map $\mathcal{F}^* \in L_{M,d}$ in four steps. Here for any $d$-cube $C \subset \mathbb{Z}^d$ which is centered at $r$, we define $V_C = [r_1, r_1+1] \times \cdots \times [r_d, r_d+1] \subset \mathbb{R}^d$.

**E**$_1$ Construct a bounded $f' \in F[A_{p,q}]$: For any $r \in A_{p,q}$ such that $l \leq r_i \leq l + n_2$ for all $1 \leq i \leq d$, we set $f'(r) = f(r')$, where $r'_i = r_i - l + 1$ for all $1 \leq i \leq d$; otherwise, letting $1 \leq i \leq d$ be the largest index such that $r_i < l$ or $r_i > l + n_2$, set $f'(r) = \text{sgn}(l - r_i)e^i$.

**E**$_2$ Construct a map $\mathcal{F}$ from $A_{p,q}$ to $\mathbb{R}^d$: $\mathcal{F}(r) = r + cf'(r)$ for all $r \in A_{p,q}$.

**E**$_3$ Use Cartesian Interpolation (for details, see Appendix G) on every $d$-cube in $A_{p,q}$. In this way, we extend $\mathcal{F}$ to be a map $\mathcal{F}'$ from $[0, n_2+2l]^d$ to $\mathbb{R}^d$ (more precisely, $\mathcal{F}'$ is a map from $[0, n_2+2l]^d$ to itself).

**E**$_4$ Construct a map $\mathcal{F}^*$ from $E^d$ to itself:

$$\mathcal{F}^*(x) = \mathcal{F}'((n_2+2l)x)/(n_2+2l), \quad \text{for all } x \in E^d.$$

Proof of Lemma 5.3 below can be found in Appendix H.

LEMMA 5.3. *Map $\mathcal{F}^*$ constructed above belongs to $L_{M,d}$. Let $x^*$ be any approximate fixed point of $\mathcal{F}^*$ with error $2^{-m}$, and $C$ be any $d$-cube in $A_{p,q}$ such that $(n_2+2l)x^* \in V_C$, then there must exist a zero point $r^* \in C$ such that $f'(r^*) = 0$, and thus, $f(r^* - l + 1) = 0$.*

Given a bounded $f \in F[N_{n_2+1,d}]$, Lemma 5.3 above implies that any algorithm for $\mathbf{AFP}^{M,d,m}$ can be used to find a zero point of $f$ as follows. First, we run the algorithm to find an approximate fixed point $x^*$ of $\mathcal{F}^*$ which is constructed above. Whenever the algorithm queries $\mathcal{F}^*(x)$ for some $x \in E^d$, we only need to evaluate $f$ at $2^d$ points and then, $\mathcal{F}^*(x)$ can be determined (see the definition of Cartesian Interpolation in Appendix G). Once the algorithm outputs an approximate fixed point $x^*$ of $\mathcal{F}^*$, $2^d$ more queries on $f$ are enough to find a zero point of $f$ according to Lemma 5.3. As $2^m M > 192d^3$ and $n_2 + 1 > 48d$, our result in Section 4 shows that $0.5\,(\lfloor n_2/2^{10} \rfloor)^{d-1} \leq 2^d Q(M, d, m) + 2^d$, which gives us the lower bound in Theorem 5.2.

## 6. *Conclusion and Remarks*

In establishing the algorithmic complexity for the discrete fixed point problem, we develop a deep lower bound proof, and a succinct algorithm for an asymptotically matching upper bound. These results allow us to close the gap between the upper and lower bounds of Hirsch et al. [1989] for the approximate fixed point problem. The novelty of our upper bound proof may shed new light on algorithm design for other related problems.

Recently, Chen and Teng [2007] studied the randomized query complexity of the discrete fixed point problem, and proved a tight lower bound of $\Omega(n^{d-1})$. Their result demonstrates that, in the query model, randomization does not help much in fixed point computation. For the quantum model, Chen et al. [2008] proved a lower bound of $\Omega(n^{(d-1)/2})$, while the upper bound is $O(n^{d/2})$.

The celebrated fixed point theorem of Brouwer followed from a concept of degree, which was also the main idea in many of his other contributions in topology. The idea can be traced back to the Kronecker Integral [Kronecker 1869]. Brouwer derived it from a discretization of the metric space under consideration, as in our definition of badness. Informally, it is the number of "positively" oriented simplices minus the number of "negatively" oriented simplices, whose images cover a given point in the range, in the limit as the simplices go to infinitely small uniformly. Brouwer proved that the value is a constant independent of the choices of the "triangulation" of the domain space. In addition, he showed that the value is invariant under homotopy, if certain conditions are satisfied. The concept of degree with these properties can be applied to derive a series of fixed point theorems. Each of them describes an interesting class of function-domain pairs which guarantees the existence of fixed points.

In particular, degree in the two-dimensional case can be simplified to the winding number. If the winding number of a function $f$ around the boundary of its domain is nonzero, it must have a fixed point. Moreover, this function-domain property defined by winding number is dividable. That is, after dividing the domain into two parts, this property still holds in one of them. Therefore, the existence of fixed point is ensured by its existence in the limit. Employing this idea, Hirsch et al. [1989] got their matching algorithmic bound for the two-dimensional case. To generalize the concept of winding number to higher dimensions, however, is not easy and has been relied on the discretization process of Brouwer's definition of degree, which is not suitable for a divide-and-conquer approach to narrow down the existence of fixed points. On the other hand, our discrete fixed point theorem describes a new class of function-domain pairs which is dividable for all dimensions. It allows us to obtain the matching algorithmic bound for arbitrary dimensions.

The oracle model used in this article is quite strong. In a more general approach, one may assume that the function is presented as a Turing machine. It becomes undecidable if the domain contains all the integer points as one can easily reduce the halting problem to it. Papadimitriou [1990] and Ko [1995] studied interesting properties of the fixed point problem for some classes of function-domain pairs, with function evaluations done by Turing machines.

Though our matching bound concludes the study of the deterministic black-box model, it still leaves room for better algorithms to be designed for specific classes of functions. There have been extensive literatures in algorithms for computing fixed points of various classes of functions [Sikorski 1989; Shellman and Sikorski 2002, 2003a,2003b; Sikorski et al. 1993; Sikorski and Wozniakowski 1987; Yang 1999]. The fixed point problem also has a strong connection with the market equilibrium problem which has been recently studied intensively on its algorithmic complexity issues [Chen et al. 2004; Codenotti and Varadarajan 2004; Deng et al. 2002, 2003; Devanur 2004; Devanur et al. 2002; Jain 2004; Jain et al. 2003, 2005]. Our results may contribute new ideas to such studies.

## *Appendix*

### A. *Proof of Lemma* 3.8

Let $C_0^{d-1} \subset \mathbb{Z}^d$ be the $(d-1)$-cube that is centered at 0 and perpendicular to $\{1\}$. We define a function $g$ on $C_0^{d-1}$ as follows: for every $r \in C_0^{d-1}$, let $i$ be the largest integer such that $r_i = 0$, then $g(r) = +e^i$.

LEMMA A.1. *$(d-1)$-cube $C_0^{d-1} \subset \mathbb{Z}^d$ is bad relative to the $g$ defined above.*

PROOF. For any $0 \le t \le d-1$, we use $r^t$ to denote the point in $C_0^{d-1}$ such that

$$r_i^t = 0 \text{ for all } 1 \le i \le t+1, \text{ and } r_i^t = 1 \text{ for all } t+2 \le i \le d.$$

For example, $r^{d-1} = 0$ and $r^0 = (0, 1, \ldots, 1)$. Let $C_*^t \subset C_0^{d-1}$ be the t-cube centered at $r^t$ and perpendicular to $\{1, t+2, t+3, \ldots, d\}$. Now we apply induction on $t$ to prove that, for any $0 \le t \le d-1$, $C_*^t$ is bad relative to $g$. Lemma A.1 then follows since $C_*^{d-1}$ is exactly $C_0^{d-1}$.

The base case for $t = 0$ is trivial. For $t \ge 1$, it is easy to check that $g(C_*^t) = \{e^1, e^2, \ldots, e^{t+1}\}$ and ($\mathbf{B}_1$) is satisfied. Note that $C_*^{t-1} \subset C_*^t$, and by the inductive hypothesis, $C_*^{t-1}$ is bad relative to $g$. For any other $(t-1)$-cube $C^{t-1} \subset C_*^t$, there must exist some point $r \in C^{t-1}$ such that $g(r) = e^{t+1}$. Condition ($\mathbf{B}_1$) is violated and $C^{t-1}$ cannot be bad. In conclusion, $C_*^{t-1}$ is the only bad $(t-1)$-cube in $C_*^t$, and ($\mathbf{B}_2$) is satisfied by $C_*^t$. As a result, $C_*^t$ is bad. □

PROOF OF LEMMA 3.8. For any $(d-1)$-cube $C^{d-1} \in V[p', q']$ centered at $r$ and perpendicular to set $\{t\}$, we prove that it is bad relative to $f'$ iff $r = p'$ and $t = 1$. Condition ($\mathbf{B}_1$) requires that $f'(C^{d-1}) = \{e^1, e^2, \ldots, e^d\}$. But if there exists $i$ such that $r_i > p'_i$, then $e^i \notin f'(C^{d-1})$. Thus, $r$ must equal to $p'$ if $C^{d-1}$ is bad.

**Algorithm Implementation of Line 5 in Cut$^d$**

```
1 : for any 0-cube $C^0 \subset S$, set its three properties according to the definition
2 : for $t = 1$ to $d-1$ do
3 :   for any $t$-cube $C^t \subset S$ do
4 :     set $C^t$.zero = $\max_{C^{t-1} \subset C^t} \{C^{t-1}$.zero$\}$ and $C^t$.max = $\max_{C^{t-1} \subset C^t} \{C^{t-1}$.max$\}$
5 :     if $C^t$.zero = 0, $C^t$.max = $t+1$ and $\sum_{C^{t-1} \subset C^t} C^{t-1}$.bad is odd, then
6 :       set $C^t$.bad = 1
7 :     else
8 :       set $C^t$.bad = 0
9 : set $V = \emptyset$
10 : for any $(d-1)$-cube $C^{d-1} \subset S$ if $C^{d-1}$.bad = 1 then $V = V \cup \{C^{d-1}\}$
```

FIG. 2. Implementation of Line 5 in **Cut**$^d$.

Under this condition, if $t > 1$, then we have $e^1 \notin f'(C^{d-1})$ which violates ($\mathbf{B}_1$). In conclusion, only the $(d-1)$-cube centered at $p'$ and perpendicular to $\{1\}$ can satisfy ($\mathbf{B}_1$). As translation does not affect the badness of cubes, this $(d-1)$-cube is bad relative to $f'$ according to Lemma A.1, and Lemma 3.8 is proven. □

## B. *Implementation of Line 5 in Cut$^d$*

The implementation of line 5 is presented in Figure 2. Here, three properties are attached to each $t$-cube $C^t \subset S$:

(1) $C^t$.bad = 1 if it is bad relative to $g$, and $C^t$.bad = 0 otherwise;

(2) $C^t$.zero = 0 if $g$ has no zero point in $C^t$, and $C^t$.zero = 1 otherwise;

(3) $C^t$.max is equal to the largest $i$ such that $+e^i \in g(C^t)$, or 0 if no such $i$ exists.

From the definition of bad cubes, it is easy to see that $C^t \subset S$, where $1 \leq t \leq d-1$, is bad iff $C^t$.zero = 0, $C^t$.max = $t+1$, and $\sum_{C^{t-1} \in C^t} C^{t-1}$.bad is odd.

By the definition, for any $r \in \mathbb{Z}^d$, there are exactly $\binom{d}{t}$ $t$-cubes centered at $r$. Thus, the number of $t$-cubes in set $S$ is at most $\binom{d}{t} \cdot |S|$. On the other hand, every $t$ cube $C^t$ contains exactly $2t$ $(t-1)$-cubes, and $O(td)$ steps are enough to enumerate all of them. As a result, the time complexity of the implementation is bounded by $O(d^2 2^d |S|)$.

## C. *Proof of Lemma 4.5*

PROOF. We use induction on $d$. The base case for $d = 1$ is trivial. For $d \geq 2$, we let $V_S = \{u \in G_{m,d} \mid (S, u) \notin K_{m,d}\}$. If $|S| \geq l_{m,d}$, then $256^{d-1} m|S| \geq m^d = |V_S|$. If $|S| = 0$, then $|V_S| = 0$, and the statement is also true. Otherwise, we assume $0 < |S| < l_{m,d}$. In accordance with the definition, we have $V_S = S \cup V^- \cup V^+$, where

$$V^- = \{u \in G_{m,d} \mid (D(S^{u_d - 1}), D(u^-)) \notin K_{m,d-1}\} \quad \text{and}$$
$$V^+ = \{u \in G_{m,d} \mid (D(S^{u_d + 1}), D(u^+)) \notin K_{m,d-1}\}.$$

Let $V = \{u \in G_{m,d} \mid (D(S^{u_d}), D(u)) \notin K_{m,d-1}\}$, then $|V^-| \leq |V|$ and $|V^+| \leq |V|$. On the other hand, $V$ can be decomposed into layers: $V = V^1 \cup V^2 \cup \cdots \cup V^m$. By the inductive hypothesis, we have $|V^t| \leq 256^{d-2}m|S^t|$ for all $1 \leq t \leq m$, and

$$|V^-| \leq |V| = |V^1| + \cdots + |V^m| \leq 256^{d-2}m|S|.$$

One can bound $|V^+|$ similarly, and the lemma is proven. □

## D. *Proof of Lemma* 4.6

LEMMA D.1. *For each graph $G_{m,d}$, we define an integer $M_{m,d}$ as*

$$\min_{(S,u)\in K_{m,d}} \big|\{w \in G_{m,d} \mid \exists \textit{ pipe path } P \textit{ starts at } u, \textit{ ends at } w, \textit{ and } P \cap S = \emptyset\}\big|.$$

*Then, we have $M_{m,1} = m$, and $M_{m,d} \geq (m-3)(M_{m,d-1} - l_{m,d})$, for any $d \geq 2$.*

PROOF. The case when $d = 1$ is trivial. For $d \geq 2$, we only focus our discussion here on the case when $2 \leq u_d \leq m-1$. The other two cases ($u_d = 1$ and $u_d = m$) are easier and can be handled similarly.

We now count the number of $w \in G_{m,d}$ that satisfies:

**F**$_1$ $w_d > u_d + 1$, and for any $v \in S$, $D(w) \neq D(v)$.
**F**$_2$ there exists a pipe path $P'$ in layer $u_d + 1$ of $G_{m,d}$ which starts at $u^+$, ends at $w[d \leftarrow (u_d+1)]$, and $P' \cap S = \emptyset$.

For every such vertex $w$, path $P$ where

$$P = uu^+ \cup P' \cup (w[d \leftarrow (u_d+1)]w[d \leftarrow (u_d+2)] \ldots w)$$

is a pipe path in $G_{m,d}$. It starts at $u$, ends at $w$, and $P \cap S = \emptyset$. The number of such $w$ is at least $(m - u_d - 1)(M_{m,d-1} - l_{m,d})$ since $|S| < l_{m,d}$.

Similarly, the number of $w$ below layer $u_d$ is at least $(u_d - 2)(M_{m,d-1} - l_{m,d})$. The lemma then follows by summing them up. □

PROOF OF LEMMA 4.6. After expanding the inequality in Lemma D.1, we get

$$M_{m,d} \geq c^{d-1}m^d\left(1 - \left(\frac{1}{256}\right)\sum_{i=0}^{d-2} b^i\right)$$

where $c = (m-3)/m$ and $b = 1/(256c)$. As $m \geq 12d \geq 12$, we have $c \geq 3/4$ and $b \leq (1/192)$. Therefore,

$$M_{m,d} \geq c^{d-1}m^d\left(1 - \left(\frac{1}{256}\right)\left(\frac{1}{1-b}\right)\right) \geq \left(\frac{761}{764}\right)c^{d-1}m^d.$$

As $m \geq 12d$, we have $c \geq 1 - 1/(4d)$, $c^{d-1} \geq 1 - (d-1)/(4d) > 3/4$, and the lemma is proven. □

## E. *Construction of Strategies for Alice*

Strategy $T[m,d]$ for Alice will be constructed inductively. When $d = 1$, it is trivial to find an $T[m,1]$ that satisfies Theorem 4.8, since $l_{m,1} = 1$ and $S = \emptyset$.

**Algorithm** $T[m,d]$.**Init**$(S,u)$

```
1:  set $Q = uu^+$ and $V = S$
2:  create a new strategy $T[m,d-1]$ on layer $u_d+1$ of $G_{m,d}$ and use $T[u_d+1]$ to denote it
3:  call $T[u_d+1]$.Init $(V^{u_d+1}, u^+)$
```

**Algorithm** $T[m,d]$.**Query**$(v)$

```
1:  Assume $Q = uv^1 \dots v^k$ and $t = v_d^k$
2:  if $v \in S$ then output false
3:  else if $v \in V$ then { $v$ must be queried at some time before } output the same answer
4:  else if $v_d > t$ then output false
5:  else if $v_d < t$ then
6:      if $v \notin Q$ then output false
7:      else output the sub-path of $Q$ that starts at $u$ and ends at the successor of $v$
8:  else if $|V^t| < (c_{m,d} - 1)$ then
9:      call $T[t]$.Query$(v)$
10:     if the output is false then output false
11:     else the output must be a path $Q^*$, output $Q \cup Q^*$
12: else { $|V^t| \geq (c_{m,d} - 1)$ }
13:     find the smallest $l$ that satisfies $l > t$ and $|V^l| < c_{m,d}$
14:     call $T[t]$.GetPaths() to get a set $R$ of pipe paths in the layer $t$ of graph $G_{m,d}$
15:     find a pipe path $Q^* \in R$, whose end $w^*$ satisfies
        (G_1). $\forall\, v \in V$, $D(w^*) \neq D(v)$   (G_2). $V^l$ is sparse relative to $w^*[d \leftarrow l]$ in layer $l$
16:     delete the strategy $T[t]$ on layer $t$ of $G_{m,d}$
17:     create a new strategy $T[m,d-1]$ on layer $l$ of $G_{m,d}$ and use $T[l]$ to denote it
18:     set $Q = Q \cup Q^* \cup (w^*w^*[d \leftarrow t+1] \dots w^*[d \leftarrow l])$ and call $T[l]$.Init$(V^l, w^*[d \leftarrow l])$
19:     if $v \notin Q$ then output false
20:     else output the sub-path of $Q$ which starts at $u$ and ends at the successor of $v$
21: set $V = V \cup \{v\}$
```

FIG. 3. Functions $T[m,d]$.**Init**$(S,u)$ and $T[m,d]$.**Query**$(v)$.

When $d \geq 2$, to build $T[m,d]$, we can assume that $T[m,d-1]$ has already been constructed (as $m \geq 12d > 12(d-1)$). Furthermore, $T[m,d-1]$ can be employed to work on any layer of $G_{m,d}$ using the isomorphic mapping $D$.

During the game of queries and answers, strategy $T[m,d]$ maintains a pipe path $Q$ in $G_{m,d}$. It starts at vertex $u$ and grows away from layer $u_d$ very slowly. There are two cases: if $u_d \leq m/2$, then $Q$ will grow from the bottom up; otherwise, $Q$ will grow from the top down. For the sake of simplicity, here we only discuss the case when $u_d \leq m/2$.

Details of $T[m,d]$ are presented in Figures 3 and 4, with $c_{m,d} \in \mathbb{R}^+$ defined as

$$c_{m,d} = \frac{l_{m,d-1}}{16} = \frac{1}{16}\left(\frac{m}{256}\right)^{d-2}.$$

During the execution, $T[m,d]$ always keeps track of set

$$V = S \cup \{v \in G_{m,d} \mid T[m,d].\mathbf{Query}(v) \text{ is called before}\}.$$

**Algorithm** $T[m,d]$.**GetPaths**()

```
1: Assume $Q = uv^1 \dots v^k$ and $t = v^k_d$
2: find the smallest $l$ such that $l > t$ and $|V^l| < c_{m,d}$
3: call $T[t]$.GetPaths() to get a set $R$ of pipe paths in the layer $t$ of graph $G_{m,d}$
4: find a pipe path $Q^* \in R$, whose end vertex $w^*$ satisfies
     (G'$_1$). $\forall\, v \in V,\ D(w^*) \neq D(v)$    (G'$_2$). $V^l$ is sparse relative to $w^*[d \leftarrow l]$ in layer $l$
5: set $Q = Q \cup Q^* \cup (w^* w^*[d \leftarrow t+1] \dots w^*[d \leftarrow l])$
6: find a set $R'$ of pipe paths in the layer $l$ of $G_{m,d}$ with distinct end vertices such that
     (G). every pipe path in $R'$ starts at $w^*[d \leftarrow l]$ and has no vertex in $V^l$
7: for any pipe path $Q' \in R'$ whose end vertex $w'$ satisfies $\forall\, v \in V,\ D(w') \neq D(v)$ do
8:   for any $l < i \le m$, output $Q \cup Q' \cup (w'w'[d \leftarrow l+1] \dots w'[d \leftarrow i])$
```

FIG. 4. Function $T[m,d]$.**GetPaths**().

**I**$_1$ $Q = uv^1 \dots v^k$ is a pipe path starting at $u$ and $Q \cap H_s = \emptyset$. (let $t = v^k_d > u_d$)

**I**$_2$ for any $i$ such that $u_d < i < t$, we have $|V^i| \ge c_{m,d}$.

**I**$_3$ for any $i$ such that $t < i \le m$, we have $H^i_s = V^i$.

**I**$_4$ a strategy $T[m,d-1]$ denoted by $T[t]$ is currently working on layer $t$ of graph $G_{m,d}$ and

- **I**$^1_4$ it is initiated by some pair in $K_{m,d-1}$ and the vertex in this pair is $v^k$;
- **I**$^2_4$ $T[t]$.**Query**$(v)$ has been called no more than $|V^t|$ and less than $c_{m,d}$ times;
- **I**$^3_4$ the forbidden set of strategy $T[t]$ at this moment is exactly the same as $H^t_s$.

**I**$_5$ every path in set $U_s$ is either a sub-path of $Q$ or equals to $Q \cup Q^*$. Here $Q^*$ is a path output by $T[t]$.**Query**$(v)$ at some time before.

FIG. 5. Invariants maintained.

It is important to note that $|V| \le |S| + l_{m,d} < 2l_{m,d}$. After the first $s$ queries, where $0 \le s \le l_{m,d}$, let $(U_s, H_s)$ be the query-answer history defined in Section 4.2. We should prove, by induction on $s$, that all the invariants in Figure 5 are maintained. We then use these invariants to derive both properties (**C**$_1$) and (**C**$_2$).

Obviously, all the invariants in Figure 5 hold when strategy $T[m,d]$ is initiated. Now we assume that after $0 \le s < l_{m,d}$ queries, all the claims are true, and then $T[m,d]$.**Query**$(v)$ is invoked again. If the branch in line 2, 3, 4, 5 or 8 is true, then it is not hard to check that all the invariants are still maintained.

We consider the case when line 12 is true. Since there are at most $|V|/c_{m,d} < (m/8)$ layers of $V$ that satisfy $|V^i| \ge c_{m,d}$, **I**$_2$ implies $t - u_d - 1 < (m/8)$ and $t \le (5m/8)$, which ensures the existence of $l$ in line 13. Using the inductive hypothesis on strategy $T[m,d-1]$ and **I**$_4$, we know that $R$ contains at least $(m^{d-1}/8)$ pipe paths in layer $t$ of graph $G_{m,d}$, which all start at $v^k$ and end at distinct vertices. Using Lemma 4.5, there are at most $256^{d-2}m|V^l| < (m^{d-1}/16)$ paths in $R$ which do not satisfy (**G**$_2$). In addition, there are at most $|V| < 2l_{m,d}$ paths in $R$ which do not satisfy (**G**$_1$). Since $2l_{m,d} + (m^{d-1}/16) < (m^{d-1}/8)$, the pipe path $Q^*$ in line 15 always exists. It is then easy to check that the way we construct $Q$ and $T[l]$ maintains all the invariants in Figure 5.

Now it is clear that $T[m,d]$ satisfies (**C**$_1$), and the only thing left is to prove that **GetPaths**() satisfies (**C**$_2$). Since all the invariants in Figure 5 are maintained, we can prove the existence of $l$ in line 2 and $Q^*$ in line 4 similarly. The way we pick $l$ guarantees that for any $i : u_d < i \neq t < l, |V^i| \geq c_{m,d}$, hence $l - u_d - 2 < (m/8)$, and $l \leq (5m/8) + 1$. By (**G**'$_2$), set $V^l$ is sparse relative to $w^*[d \leftarrow l]$ in the layer $l$ of $G_{m,d}$. So by using Lemma 4.6, we have $|R'| \geq m^{d-1}/2$. Therefore, the number of pipe paths output by **GetPaths**() is at least

$$(m - l)(|R'| - |V|) > m^d/8.$$

Properties in ($\mathbf{C}_2$) are all satisfied and Theorem 4.8 is proven.

## F. *Construction of Functions from Pipe Paths*

In this section, we describe a method that, given any pipe path $P$ in graph $G_{m,d}$, constructs a bounded function $f_P \in F[N_{n,d}]$, where $n = 4m + 1$.

First, we define a set $I_P \subset N_{n,d}$, which looks like a pipe. It consists of two parts, kernel $K_P$ and boundary $B_P$. After defining $f_P$ on $I_P$, we extend it onto $N_{n,d}$ and prove that it is both bounded and direction-preserving. Finally, we prove properties ($\mathbf{D}_1$) and ($\mathbf{D}_2$). To clarify the presentation here, we always use $u, v, w$ to denote vertices in $G_{m,d}$, and $p, q, r$ to denote points in $N_{n,d}$.

We start with some notation. We use $T$ to denote the following map from the vertex set of $G_{m,d}$ to $N_{n,d}$: $T(v) = r$, where $r_i = 4v_i - 1$ for all $1 \leq i \leq d$. For any edge $uv \in G_{m,d}$, we use $E(uv) \subset N_{n,d}$ to denote the set of five points on segment $T(u)T(v)$.

*Definition* F.1. For any path $P = v^1 \dots v^k$ in $G_{m,d}$, we define set $I_P = K_P \cup B_P$, where $K_P = \cup_{i=1}^{k-1} E(v^i v^{i+1})$, and

$$B_P = \{r \in N_{n,d} \text{ and } r \notin K_P \mid \exists\, r' \in K_P, |r - r'|_\infty = 1\}.$$

When $k = 1$, we use $f_{\{v^1\}}, K_{\{v^1\}}, B_{\{v^1\}}$ and $I_{\{v^1\}}$ to denote $f_P, K_P, B_P$ and $I_P$.

We first describe an inductive method that, given a pipe path $P = u \dots w$ in $G_{m,d}$, constructs a function $f_P \in F[I_P]$. Let $\overline{P}$ denote the reversion of path $P$, then the method satisfies:

$\mathbf{H}_1$ For any point $r \in B_P$, $f_P(r) = f_{\overline{P}}(r)$; and

$\mathbf{H}_2$ $f_P(T(w)) = 0$; for every other point $r \in K_P$, $f_P(r) = \text{sgn}(w_d - u_d)e^d$.

As a result, we only need to describe the method for pipe paths such that $u_d \leq w_d$ (when $u_d > w_d$, we can use $f_{\overline{P}}$, together with properties ($\mathbf{H}_1$) ($\mathbf{H}_2$), to build $f_P$).

Suppose $P = u \dots w$ and $a = u_d \leq w_d = b$. Let $p = T(u)$ and $q = T(w)$, then $p_d = 4a - 1, q_d = 4b - 1$ and $a \leq b$. The method for the case when $d = 1$ is easy. As $B_P = \{p^-, q^+\}$, we simply set $f_P(p^-) = +e^1$ and $f_P(q^+) = -e^1$.

For the case when $d \geq 2$, we may assume for any pipe path $Q$ in $G_{m,d-1}$, $f_Q$ on $I_Q \subset N_{n,d-1}$ has already been constructed. According to the definition of pipe paths, for any $a \leq t \leq b$, $Q_t = D(P^t)$ is a pipe path in $G_{m,d-1}$. Let $v^t \in G_{m,d-1}$ be the start vertex of path $Q_t$, then we have $Q_a = D(u) = v^a = v^{a+1}$, $Q_b = D(w) = v^b$ and for every $a < t < b$, $Q_t$ starts at $v^t$ and ends at $v^{t+1}$. Using the inductive hypothesis, we may assume that $f_{Q_t}$, $f_{\overline{Q}_t}$ and $f_{\{v^t\}}$ have already been constructed, and they will be used to build our function $f_P$.

For any point $r \in B_P$, we will use $D$ to map it to $N_{n,d-1}$, and give it value there using one of $f_{Q_t}$, $f_{\overline{Q}_t}$ or $f_{\{v^t\}}$. Before that, we analyze set $D(B^t_P) = D(I^t_P) - D(K^t_P)$

**Value of $f_P(r)$ for any $r \in I_P$**

```
 1: if $r = q$ , then output 0
 2: else if $r \in K_P$ , then output $+e^d$
 3: else if $r = p^-$ , then output $+e^d$
 4: else if $r = q^+$ , then output $-e^d$
 5: else if $r_d = 4a-2, 4a-1$ or $4a$ , then output $f_{\{v^a\}}(D(r))$
 6: else if $r_d = 4b-2, 4b-1$ or $4b$ , then output $f_{\{v^b\}}(D(r))$
 7: else if $r_d = 4i+1$ where $a \le i < b$, then output $f_{\{v^{i+1}\}}(D(r))$
 8: else if $r_d = 4i-1$ where $a < i < b$, then output $f_{Q_i}(D(r)) = f_{\overline{Q}_i}(D(r))$
 9: else if $r_d = 4i-2$ where $a < i < b$, then output $f_{\overline{Q}_i}(D(r))$
10: else if $r_d = 4i$ where $a < i < b$, then output $f_{Q_i}(D(r))$
```

FIG. 6. Construction of function $f_P$ on $I_P \subset N_{n,d}$.

for each $t$:

(1) $t = 4a-2$, then $K_P^t = \emptyset$ and $D(I_P^t) = I_{\{v^a\}}$
(2) $t = 4b$, then $K_P^t = \emptyset$ and $D(I_P^t) = I_{\{v^b\}}$
(3) $t = 4i-1$, where $a \le i \le b$, then $D(K_P^t) = K_{Q_i}$ and $D(I_P^t) = I_{Q_i}$
(4) $t = 4i-2$, where $a < i \le b$, then $D(K_P^t) = K_{\{v^i\}}$ and $D(I_P^t) = I_{Q_i}$
(5) $t = 4i$, where $a \le i < b$, then $D(K_P^t) = K_{\{v^{i+1}\}}$ and $D(I_P^t) = I_{Q_i}$
(6) $t = 4i+1$, where $a \le i < b$, then $D(K_P^t) = K_{\{v^{i+1}\}}$ and $D(I_P^t) = I_{\{v^{i+1}\}}$

The method is described in Figure 6. It is easy to prove the following properties:

(**H**$_3$) For any $r \in B_P$ except $p^-$ and $q^+$, $f_P(r) \notin \{\pm e^d\}$;
(**H**$_4$) For any $r \in I_P$ except $q = T(w)$, $f_P(r) \neq 0$;
(**H**$_5$) For any $r \in I_{\{w\}}$, $f_{\{w\}}(r) = f_P(r)$.

Using these properties, we have

LEMMA F.2. *For any pipe path $P$ in $G_{m,d}$, $f_P$ is direction-preserving on $I_P$.*

PROOF. We use induction on $d$. The base case for $d = 1$ is trivial. For $d \ge 2$, we may assume that for any pipe path $Q$ in graph $G_{m,d-1}$, $f_Q$ is direction-preserving on $I_Q$. We now prove for any $r^1, r^2 \in I_P$ such that $|r^1 - r^2|_\infty = 1$,

$$|f_P(r^1) - f_P(r^2)|_\infty \le 1. \tag{1}$$

If one of these two points belongs to set $K_P \cup \{p^-, q^+\}$, then (1) follows directly from properties (**H**$_2$) and (**H**$_3$). Otherwise, both $f_P(r^1)$ and $f_P(r^2)$ are determined by lines 5–10 of Figure 6. Clearly, if $r_d^1 = r_d^2$, then (1) follows from the inductive hypothesis. As a result, we only need to consider the case in which $r^1, r^2 \notin K_P \cup \{p^-, q^+\}$ and $r_d^1 \neq r_d^2$.

Without loss of generality, assume $r_d^2 = r_d^1 + 1$. Let $r^3 = (r^1)^+$ and $r^4 = (r^2)^-$. We now prove that, one of following two statements must be true: (**A**) $f_P(r^1) = f_P(r^3)$; (**B**) $f_P(r^2) = f_P(r^4)$. Four different cases are discussed below:

$r_d^1 = 4i-1$, then (**H$_1$**) $\Rightarrow$ (**A**) $r_d^1 = 4i$, then (**H$_5$**) $\Rightarrow$ (**B**)
$r_d^1 = 4i+1$, then (**H$_5$**) $\Rightarrow$ (**A**) $r_d^1 = 4i-2$, then (**H$_1$**) $\Rightarrow$ (**B**)

```
Value of $f_P(r)$ for any $r \in N_{n,d}$ and $r \notin I_P$
1: if $r_d = p_d$, then output $f_{\{v\}}(D(r))$ { Here $p = T(u)$ and $v = D(u)$ }
2: else if $r_d < p_d$, then output $+e^d$
3: else if $r_d > p_d$, then output $-e^d$
```

FIG. 7. Extension of $f_P$ from $I_P$ onto $N_{n,d}$.

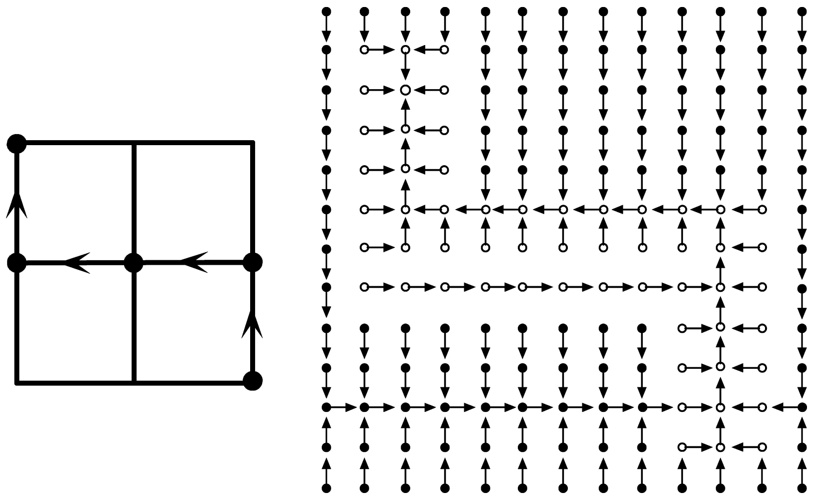

FIG. 8. A pipe path $P$ in $G_{3,2}$ and $f_P$ on $N_{13,2}$.

If (**A**) is true, then $|f_P(r^1) - f_P(r^2)|_\infty = |f_P(r^3) - f_P(r^2)|_\infty \leq 1$. The last inequality follows from the fact that $r_d^3 = r_d^2$ and $|r^3 - r^2|_\infty \leq 1$. The case when (**B**) is true can be proved similarly. □

The following property of our method can be proved by induction on $d$.

LEMMA F.3. *Let $P_1$ and $P_2$ be two pipe paths in $G_{m,d}$ that both start with $P' = v^1 \dots v^k$ where $k > 1$, then for every point $r \in I_{P^*}$ where $P^* = v^1 \dots v^{k-1}$, we have $f_{P_1}(r) = f_{P_2}(r)$.*

We now inductively extend the function $f_P \in F[I_P]$ onto $N_{n,d}$. The method for case $d = 1$ is easy. For any $r \in N_{n,1}$ and $r \notin I_P$, we set $f_P(r) = \text{sgn}(p_1 - r_1)e^1$, where $p = T(u)$ and $u$ is the start vertex of $P$. For $d \geq 2$, we may assume that $f_{\{v\}}$ on $N_{n,d-1}$, where $v = D(u)$, has already been constructed. Figure 7 shows the way to extend $f_P$, and Figure 8 gives an example of the case when $d = 2$. Using Lemma F.2, Lemma F.4 can be proved easily by induction on $d$.

LEMMA F.4. *For any pipe path $P$ in graph $G_{m,d}$, function $f_P$ is both bounded and direction-preserving on $N_{n,d}$.*

Finally, we prove properties (**D**$_1$) and (**D**$_2$) in Section 4.3. We first define map $g$ as follows.

*Definition* F.5. For any vertex $v$ in $G_{m,d}$, we define $T_V(v) \subset N_{n,d}$ as

$$T_V(v) = \{r \in N_{n,d} \text{ and } |r - T(v)|_\infty \leq 1\}.$$

**Evaluation of $f_P(r)$ using at most two queries**

```
 1: if there exists a vertex $v \in G_{m,d}$ such that $r \in T_V(v)$ then $\{g(r) = (v, v)\}$
 2:    query Alex the vertex $v$
 3:    if the answer is false, then $\{ r \notin I_P \}$ output $f_P(r)$ using Figure 7
 4:    else { the answer must be a path $Q$ } extend $Q$ to be a pipe path $R$, output $f_R(r)$
 5: else if there exists $uv \in G_{m,d}$ such that $r \in T_E(uv)$ then $\{g(r) = (u, v)\}$
 6:    query Alex both $u$ and $v$
 7:    if one of the answers is false, then $\{ r \notin I_P \}$ output $f_P(r)$ using Figure 7
 8:    else { the answers must be two paths, let the longer one be $Q$ }
 9:       if edge $uv \notin Q$, then $\{ r \notin I_P \}$ output $f_P(r)$ using Figure 7
10:       else extend $Q$ to be a pipe path $R$ and output $f_R(r)$
11: else { $g(r)$ = (“null”, “null”) and $r \notin I_P$ } output $f_P(r)$ using Figure 7
```

FIG. 9. Evaluation of $f_P(r)$.

For any edge $uv$ in $G_{m,d}$, let $r' = (T(u) + T(v))/2 \in N_{n,d}$, and $i$ be the index such that $|u_i - v_i| = 1$, then we define $T_E(uv) \subset N_{n,d}$ as

$$T_E(uv) = \{r \in N_{n,d},\ r_i = r'_i \text{ and } |r - r'|_\infty \leq 1\}.$$

Obviously, all the subsets of $N_{n,d}$ defined above are pairwise disjoint.

For $r \in N_{n,d}$, if there exists a vertex $v$ in $G_{m,d}$ such that $r \in T_V(v)$, then we set $g(r) = (v, v)$; if there exist two vertices $u, v$ in $G_{m,d}$ such that $r \in T_E(uv)$, then $g(r) = (u, v)$; otherwise, $g(r) =$ (“null”, “null”). Property (**D**$_1$) is obvious, as $f_P(r) = 0$ if and only if $r = q = T(w)$, where $w$ is the ending vertex of $P$.

The following two properties are both easy to check.

LEMMA F.6. *For every path $P = v^1 \ldots v^k$ in $G_{m,d}$, we have*

$$I_P = \big( \cup_{i=1}^{k} T_V(v^i) \big) \cup \big( \cup_{i=1}^{k-1} T_E(v^i v^{i+1}) \big).$$

LEMMA F.7. *For every pipe path $P$ in $G_{m,d}$ and $r \in N_{n,d}$,*

(1) *if there exists a vertex $v$ in $G_{m,d}$ such that $r \in T_V(v)$, then $r \in I_P$ iff $v \in P$;*
(2) *if there exists an edge $uv$ in $G_{m,d}$ such that $r \in T_E(uv)$, then $r \in I_P$ iff $uv \in P$;*
(3) *otherwise, $r \notin I_P$.*

Finally, we prove property (**D**$_2$). Let $P$ be the pipe path held by Alex. Nothing is known about $P$ except its start vertex $u \in G_{m,d}$. For any $r \in N_{n,d}$, to decide $f_P(r)$, Bob only need to query Alex for $v_1$ and $v_2$, where $(v_1, v_2) = g(v)$. Clearly, if one of the answers is (“true”, $P$), then Bob can decide $f_P(r)$ easily, since the whole path $P$ is revealed to him. Otherwise, Bob can use Figure 9 to decide $f_P(r)$. Lemma F.6, Lemma F.7, and Lemma F.3 together guarantee that the output of Figure 9 is exactly $f_P(r)$.

## G. *Definition and Properties of Cartesian Interpolation*

For any $d$-cube $C \subset \mathbb{Z}^d$ (since only $d$-cubes are considered here, we use $C$ instead of $C^d$) that is centered at $r$, we define set $V_C \subset \mathbb{R}^d$ as $[r_1, r_1+1] \times \cdots \times [r_d, r_d+1]$.

For any map $\mathcal{F}$ from $C$ to $\mathbb{R}^d$, the Cartesian Interpolation extends it to be a map $\mathcal{F}'$ from $V_C$ to $\mathbb{R}^d$.

*Definition* G.1. For any $r \in C$, function $w_r$ from $V_C$ to $[0, 1]$ is defined as

$$w_r(x) = \prod_{i=1}^{d}(1 - |x_i - r_i|).$$

Using $w_r$, we extend $\mathcal{F}$ to be a map $\mathcal{F}'$ from $V_C$ to $\mathbb{R}^d$:

$$\mathcal{F}'(x) - x = \sum_{r \in C} w_r(x)(\mathcal{F}(r) - r), \quad \text{for all } x \in V_C.$$

One can check that for any $x \in V_C$, $\sum_{r\in C} w_r(x) = 1$. Lemma G.2 below is easy to prove.

LEMMA G.2. *If for any $r \in C$, $\mathcal{F}$ satisfies $a \le \mathcal{F}_i(r) \le b$, then for any $x \in V_C$, $a \le \mathcal{F}'_i(x) \le b$.*

LEMMA G.3. *If for any $r \in C$, $\mathcal{F}$ satisfies $|\mathcal{F}(r) - r|_\infty \le L$, then $\mathcal{F}'(x) - x$ on $V_C$ satisfies a Lipschitz condition with constant $2dL$.*

PROOF. We only need to prove for any $i : 1 \le i \le d$, $g(x) = \mathcal{F}'_i(x) - x_i$ satisfies

$$|g(x) - g(y)| \le 2dL|x - y|_\infty, \quad \text{for all } x, y \in V_C.$$

For each integer $0 \le j \le d$, we define a point $z^j \in V_C$ as $z^0 = y$, $z^j = (x_1, \ldots, x_j, y_{j+1}, \ldots, y_d)$, where $1 \le j \le d-1$, and $z^d = x$. Then

$$|g(x) - g(y)| = |g(z^d) - g(z^0)| \le \sum_{j=0}^{d-1} |g(z^j) - g(z^{j+1})|.$$

For each $0 \le j < d$, it is easy to check that

$$\sum_{r\in C} |w_r(z^j) - w_r(z^{j+1})| = 2|x_{j+1} - y_{j+1}| \le 2|x - y|_\infty.$$

Therefore, for every $0 \le j < d$, we have

$$|g(z^j) - g(z^{j+1})| \le \sum_{r\in C} |w_r(z^j) - w_r(z^{j+1})| \cdot |\mathcal{F}_i(r) - r_i| \le 2L|x - y|_\infty.$$

As a result, $|g(x) - g(y)| \le 2dL\,|x - y|_\infty$, and the lemma is proved. □

Let $\mathcal{F}$ be a map from $A_{p,q}$ to $\mathbb{R}^d$, then we can apply Cartesian Interpolation on every $d$-cube in $A_{p,q}$. Let $C_1$ and $C_2$ be two $d$-cubes in $A_{p,q}$, and $x \in V_{C_1} \cap V_{C_2}$. Then the $\mathcal{F}'(x)$ interpolated from $C_1$ is exactly the same as the $\mathcal{F}'(x)$ interpolated from $C_2$. In this way, we can extend $\mathcal{F}$ to be a map $\mathcal{F}'$ from $V = [p_1, q_1] \times [p_2, q_2] \times \cdots \times [p_d, q_d]$ to $\mathbb{R}^d$, which has the following Lipschitz property.

LEMMA G.4. *If for any $r \in A_{p,q}$, $\mathcal{F}$ satisfies $|\mathcal{F}(r) - r|_\infty \le L$, then $\mathcal{F}'(x) - x$ on $V$ defined above satisfies a Lipschitz condition with constant $2dL$.*

PROOF. Let $\mathcal{G}(x) = \mathcal{F}(x) - x$. We only need to prove for any $x, y \in V$,

$$|\mathcal{G}(x) - \mathcal{G}(y)|_\infty \le 2dL|x - y|_\infty.$$

If there exists a $d$-cube $C \subset A_{p,q}$ such that $x, y \in V_C$, then the lemma follows from Lemma G.3. Otherwise, we can divide the segment $xy$ into $x^0x^1, x^1x^2, \ldots, x^{k-1}x^k$ such that $x^0 = x, x^k = y$, and for every $0 \le i < k$, there exists a $d$-cube $C_i \subset A_{p,q}$ that satisfies $x^i, x^{i+1} \in V_{C_i}$. Using Lemma G.3, we have

$$|\mathcal{G}(x) - \mathcal{G}(y)|_\infty \le \sum_{i=0}^{k-1} |\mathcal{G}(x^i) - \mathcal{G}(x^{i+1})|_\infty \le 2dL \sum_{i=0}^{k-1} |x^i - x^{i+1}|_\infty = 2dL|x-y|_\infty,$$

and the lemma is proven. □

## H. *Proof of Lemma* 5.3

PROOF. Let $V$ be $[0, n_2 + 2l]^d$, then the way we build $\mathcal{F}$ guarantees that it is a map from $A_{p,q}$ to $V$. It then follows from Lemma G.2 that $\mathcal{F}'$ is a map from $V$ to itself, and thus $\mathcal{F}^*$ is a map from $E^d$ to itself. Since $|\mathcal{F}(r) - r|_\infty \le c$ for all point $r \in A_{p,q}$, Lemma G.4 shows that $\mathcal{F}'(x) - x$ satisfies a Lipschitz condition with constant $2dc = M$. After the scaling operation in step $\mathbf{E}_4$, it is easy to check that $\mathcal{F}^*(x) - x$ satisfies the same Lipschitz property, and we have $\mathcal{F}^* \in L_{M,d}$.

Now we prove the second part of Lemma 5.3. Suppose there is no zero point $r^*$ in $C$ such that $f'(r^*) = 0$, then for each index $1 \le i \le d$, we define

$$V_i = \{r \in C \mid f'(r) = +e^i \text{ or } -e^i\} \text{ and } a_i = \sum_{r \in V_i} w_r(x'),$$

where $x' = (n_2 + 2l)x^*$. Because $\sum_{i=1}^d a_i = 1$, there exists an index $k$ such that $a_k \ge 1/d$. On the other hand, since $f'$ is direction-preserving, all the points $r \in V_k$ must have the same $f'(r)$. Without loss of generality, we assume $f'(r) = +e^k$ for all $r \in V_k$, then

$$\mathcal{F}'_k(x') - x'_k = \sum_{r \in C} w_r(x')(\mathcal{F}_k(r) - r_k) = a_k c \ge M/(2d^2),$$

which contradicts with

$$|\mathcal{F}'_k(x') - x'_k| \le |\mathcal{F}'(x') - x'|_\infty = (n_2 + 2l) \cdot |\mathcal{F}^*(x^*) - x^*|_\infty < M/(2d^2),$$

as we assumed that $2^m > 4d$. □

ACKNOWLEDGMENTS. The authors would like to give thanks to Frances Yao for many research discussion sessions with us; to Frances and Andrew Chi-Chih Yao for patiently spending hours going through the proofs; to Yinyu Ye, Chuangyin Dang, and Shang-Hua Teng for their suggestions and comments; and to Stephen Vavasis for references. We would also like to thank several anonymous referees of STOC 2005 and JACM for valuable comments, references and suggestions that have helped to improve the presentation of the journal version.

## REFERENCES

ALTMAN, E., AVRACHENKOV, K., AND BARAKAT, C. 2002. Tcp network calculus: The case of large delay-bandwidth product. In *Proceedings of the 21st Annual Joint Conference of the IEEE Computer and Communications Societies*. IEEE, Computer Society Press, Los Alamitos, CA, 417–426.

ARROW, K., AND DEBREU, G. 1954. Existence of an equilibrium for a competitive economy. *Econometrica 22*, 265–290.

BANACH, S. 1922. Sur les opérations dans les ensembles abstraits et leur application aux Équations intégrales. *Fund. Math. 3*, 133–181.

BROUWER, L. 1910. Über abbildung von mannigfaltigkeiten. *Math. Ann. 71*, 97–115.

CHEN, N., DENG, X., SUN, X., AND YAO, A. C.-C. 2004. Fisher equilibrium price with a class of concave utility functions. In *Proceedings of the 12th Annual European Symposium on Algorithms*. Springer-Verlag, New York, 169–179.

CHEN, X., SUN, X., AND TENG, S.-H. 2008. Quantum separation of local search and fixed point computation. In *Proceedings of the 14th Annual International Computing and Combinatorics Conference*. Springer-Verlag, New York, 169–178.

CHEN, X., AND TENG, S.-H. 2007. Paths beyond local search: A tight bound for randomized fixed-point computation. In *Proceedings of the 48th Annual IEEE Symposium on Foundations of Computer Science*. IEEE, Computer Society Press, Los Alamitos, CA, 124–134.

CODENOTTI, B., AND VARADARAJAN, K. 2004. Efficient computation of equilibrium prices for markets with Leontief utilities. In *Proceedings of the 31st International Colloquium on Automata, Languages and Programming*. Springer-Verlag, New York, 371–382.

COLE, R., DODIS, Y., AND ROUGHGARDEN, T. 2003. Pricing network edges for heterogeneous selfish users. In *Proceedings of the 35th Annual ACM Symposium on Theory of Computing*. ACM, New York, 521–530.

DENG, X., PAPADIMITRIOU, C., AND SAFRA, S. 2002. On the complexity of equilibria. In *Proceedings of the 34th Annual ACM Symposium on Theory of Computing*. ACM, New York, 67–71.

DENG, X., PAPADIMITRIOU, C., AND SAFRA, S. 2003. On the complexity of price equilibria. *J. Comput. Syst. Sci. 67*, 2, 311–324.

DEVANUR, N. 2004. The spending constraint model for market equilibrium: Algorithmic, existence and uniqueness results. In *Proceedings of the 36th Annual ACM Symposium on Theory of Computing*. ACM, New York, 519–528.

DEVANUR, N., PAPADIMITRIOU, C., SABERI, A., AND VAZIRANI, V. 2002. Market equilibrium via a primal-dual-type algorithm. In *Proceedings of the 43rd Annual IEEE Symposium on Foundations of Computer Science*. IEEE, Computer Society Press, Los Alamitos, CA, 389–395.

EAVES, B. 1972. Homotopies for computation of fixed points. *Math. Prog. 3*, 1–22.

HIRSCH, M., PAPADIMITRIOU, C., AND VAVASIS, S. 1989. Exponential lower bounds for finding Brouwer fixed points. *J. Complex. 5*, 379–416.

HUANG, Z., KHACHIYAN, L., AND SIKORSKI, K. 1999. Approximating fixed points of weakly contracting mappings. *J. Complex. 15*, 200–213.

IIMURA, T. 2003. A discrete fixed point theorem and its applications. *J. Math. Econ. 39*, 7, 725–742.

IIMURA, T., MUROTA, K., AND TAMURA, A. 2005. Discrete fixed point theorem reconsidered. *J. Math. Econ. 41*, 8, 1030–1036.

JAIN, K. 2004. A polynomial time algorithm for computing an Arrow-Debreu market equilibrium for linear utilities. In *Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science*. IEEE, Computer Society Press, Los Alamitos, CA, 286–294.

JAIN, K., MAHDIAN, M., AND SABERI, A. 2003. Approximating market equilibria. In *Proceedings of the 6th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems and the 7th International Workshop on Randomization and Approximation Techniques in Computer Science*. Springer-Verlag, New York, 861–869.

JAIN, K., VAZIRANI, V., AND YE, Y. 2005. Market equilibria for homothetic, quasi-concave utilities and economies of scale in production. In *Proceedings of the 16th Annual ACM-SIAM Symposium on Discrete Algorithms*. SIAM, Philadelphia, PA, 63–71.

KELLOGG, R., LI, T., AND YORKE, J. 1976. Constructive proof of the Brouwer fixed point theorem and computational results. *SIAM J. Numer. Anal. 13*, 473–483.

KO, K.-I. 1995. Computational complexity of fixed points and intersection points. *J. Complex. 11*, 265–292.

KRONECKER, L. 1869. Über systeme von funktionen mehrerer variabeln. *Monatsber. Berlin Akad.*, 159–193 and 688–698.

KUHN, H. 1968. Simplicial approximation of fixed points. *Proc. Nat. Acad. Sci. 61*, 1238–1242.

LOW, S. 2003. A duality model of tcp and queue management algorithms. *IEEE/ACM Trans. Netw. 11*, 4, 525–536.

MEHTA, A., SHENKER, S., AND VAZIRANI, V. 2003. Profit-maximizing multicast pricing by approximating fixed points. In *Proceedings of the 4th ACM Conference on Electronic Commerce*. ACM, New York, 218–219.

MEINARDUS, G. 1963. Invarianz bei linearen approximationen. *Arch. Rational. Mech. Anal. 14*, 1, 301–303.

MERRILL, O. 1972. *Applications and Extensions of an Algorithm that Computes Fixed Points of Certain Upper Semi-Continuous Point-to-Set Mappings*. Ph.d. dissertation, University of Michigan, Ann Arbor, MI.

NASH, J. F. 1950. Equilibrium points in n-person games. *Proc. Nat. Acad. Sci. USA 36*, 48–49.

ORTEGA, J., AND RHEINBOLT, W. 1970. *Iterative Solution of Nonlinear Equations in Several Variables*. Academic Press, New York.

PAPADIMITRIOU, C. 1990. On graph-theoretic lemmata and complexity classes. In *Proceedings of the 31st Annual IEEE Symposium on Foundations of Computer Science*. IEEE, Computer Society Press, Los Alamitos, CA, 794–801.

ROBINSON, C. 1999. *Dynamical Systems, Stability, Symbolic Dynamics, and Chaos*. CRC Press.

SCARF, H. 1967. The approximation of fixed points of a continuous mapping. *SIAM J. Applied Mathematics 15*, 997–1007.

SHELLMAN, S., AND SIKORSKI, K. 2002. A two-dimensional bisection envelope algorithm for fixed points. *J. Complex. 18*, 2, 641–659.

SHELLMAN, S., AND SIKORSKI, K. 2003a. Algorithm 825: A deep-cut bisection envelope algorithm for fixed points. *ACM Trans. Math. Softw. 29*, 3, 309–325.

SHELLMAN, S., AND SIKORSKI, K. 2003b. A recursive algorithm for the infinity-norm fixed point problem. *J. Complexity 19*, 6, 799–834.

SIKORSKI, K. 1989. Fast algorithms for the computation of fixed points. In *Robustness in Identification and Control*, R. Milanese and A. Vicino, Eds. Plenum Press, New York, 49–59.

SIKORSKI, K., TSAY, C., AND WOZNIAKOWSKI, H. 1993. An ellipsoid algorithm for the computation of fixed points. *J. Complex. 9*, 1, 181–200.

SIKORSKI, K., AND WOZNIAKOWSKI, H. 1987. Complexity of fixed points, I. *J. Complex. 3*, 4, 388–405.

SMALE, S. 1976. A convergent process of price adjustment and global newton methods. *J. Math. Econ. 3*, 2, 107–120.

SPERNER, E. 1928. Neuer beweis fur die invarianz der dimensionszahl und des gebietes. *Abhandlungen aus dem Mathematischen Seminar Universitat Hamburg 6*, 265–272.

SPIELMAN, D., AND TENG, S.-H. 1996. Spectral partitioning works: Planar graphs and finite element meshes. In *Proceedings of the 37th Annual IEEE Symposium on Foundations of Computer Science*. IEEE, Computer Society Press, Los Alamitos, CA, 96–105.

YANG, Z. 1999. *Computing equilibria and fixed points: The solution of nonlinear inequalities*. Kluwer Academic Publishers, Dordrecht.

RECEIVED MARCH 2005; REVISED AUGUST 2005 AND MAY 2006; ACCEPTED MAY 2008