Computational Models and Complexities of Tarski's Fixed Points*

Chuangyin Dang
Dept. of Systems Engineering & Engineering Management
City University of Hong Kong
Kowloon, Hong Kong
E-Mail: mecdang@cityu.edu.hk

Qi Qi
Dept. of Management Science & Engineering
Stanford University
Stanford, CA 94305-4026
E-Mail: kaylaqi@stanford.edu

Yinyu Ye
Dept. of Management Science & Engineering
Stanford University
Stanford, CA 94305-4026
E-Mail: yinyu-ye@stanford.edu

Abstract

We consider two models of computation for Tarski's order preserving function f related to fixed points in a complete lattice: the oracle function model and the polynomial function model. We develop a complete understanding under the oracle function model for finding a Tarski's fixed point as well as determining the uniqueness of Tarski's fixed point in both the lexicographic ordering and the componentwise ordering lattices. Moreover, we present a polynomial-time reduction of an integer program to an order preserving mapping f from a lattice L into itself. As a result of this reduction, we prove that, when f is given as a polynomial function, determining whether or not f has a unique fixed point is *Co-NP hard*.

Keywords: Lexicographic Ordering, Componentwise Ordering, Lattice, Finite Lattice, Order Preserving Mapping, Fixed Point, Integer Programming, Co-NP Completeness, Co-NP Hardness

1 Introduction

A partially order set L is defined with $\preceq$ as a binary relation on the set L such that $\preceq$ is reflexive, transitive, and anti-symmetric. A lattice is a partially ordered set $(L, \preceq)$, in which any two elements

*This work was partially supported by GRF: CityU 113308 of the Government of Hong Kong SAR.

$\mathbf{x}$ and $\mathbf{y}$ have a least upper bound (supremum), $\sup_L(\mathbf{x}, \mathbf{y}) = \inf\{\mathbf{z} \in L \mid \mathbf{x} \preceq \mathbf{z} \text{ and } \mathbf{y} \preceq \mathbf{z}\}$, and a greatest lower bound (infimum), $\inf_L(\mathbf{x}, \mathbf{y}) = \sup\{\mathbf{z} \in L \mid \mathbf{z} \preceq \mathbf{x} \text{ and } \mathbf{z} \preceq \mathbf{y}\}$, in the set. A lattice $(L, \preceq)$ is complete if every nonempty subset of L has a supremum and an infimum in L. Let f be a mapping from L to itself. f is order-preserving if $f(\mathbf{x}) \preceq f(\mathbf{y})$ for any $\mathbf{x}$ and $\mathbf{y}$ of L with $\mathbf{x} \preceq \mathbf{y}$.

The well-known Tarski's fixed point theorem (Tarski)[23] asserts that, if $(L, \preceq)$ is a complete lattice and f is order-preserving from L into itself, then there exists some $\mathbf{x}^* \in L$ such that $f(\mathbf{x}^*) = \mathbf{x}^*$. This theorem plays a crucial role in the study of supermodular games (or games with strategic complementarities) for economic analysis. Supermodular games were formalized in Topkis [24] and have been extensively applied in the literature such as Bernstein and Federgruen [2][3], Cachon [6], Cachon and Lariviere [7], Fudenberg and Tirole [15], Lippman and McCardle [19], Milgrom and Roberts [20][21], Milgrom and Shannon [22], Topkis [25], and Vives [26][27]. To compute a Nash equilibrium of a supermodular game, a generic approach is to convert it into the computation of a fixed point of an order preserving mapping. Recently, an algorithm has been proposed in Echenique [14] to find all pure strategy Nash equilibria of a supermodular game, which motivated to the study in this paper.

An efficient computational algorithm for the fixed point has been a recognized important technical advantage in applications. Further, it is sometimes desirable to know if an already-found fixed point for such applications is unique or not, for the decision whether additional resource should be spent to improve the already found solution. There were some interesting complexity results in algorithmic game theory research along this line, on determining whether or not a game has a unique equilibrium point. For the bimatrix game, Gilboa and Zemel [16] showed that it is NP-hard to determine whether or not there is a second Nash equilibrium. For this problem, computing even one equilibrium (which is know to exist), is already difficult and no polynomial time algorithms are known: Nash equilibrium for the bimatrix game is known to be PPAD-complete [11]. Similar cases are known for other problems such as the market equilibrium computation (Codenotti et al.)[5].

In this work, we consider the fixed point computation of order preserving functions over a complete lattice, both for finding a solution and for determining the uniqueness of an already-found solution. We are interested in both the oracle function model and polynomial function model. For the fixed point problem, the domain space is usually huge. As we are considering a discrete version with the lattice, the search space is well-defined. For an instance of the problem, we would need an order preserving function and functions for lattice operations. For those functions, one or two lattice points are given as the input and another lattice point will be returned as the output. Therefore, the most succinct representation of the lattice $(L, \preceq)$ will be those enough to represent a lattice point which requires $O(\log |L|)$ bits. That observation has been the main motivation in most interesting discussions restricting the input size to $\log |L|$. It is enough for the representation of a variable in a lattice of size $|L|$. Both the oracle function model and the polynomial time function model return the function value $f(x)$ on a lattice node x where x is of size $\log |L|$. They differ in the ways the functions are computed. The polynomial time function model computes $f(x)$ by an explicitly given algorithm, in time polynomial of $\log |L|$. The oracle model, on the other hand, always returns the value in one oracle step. More details on comparing the two models can be found in Section 2.2.

1.1 Main Results

We focus on the componentwise ordering and lexicographic ordering finite lattices. Let $L_d = \{\mathbf{x} \in Z^d \mid \mathbf{a} \leq \mathbf{x} \leq \mathbf{b}\}$, where $\mathbf{a}$ and $\mathbf{b}$ are two finite vectors of Z^d with $\mathbf{a} < \mathbf{b}$. We denote the compo-

nentwise ordering and the lexicographic ordering as $\leq_c$ and $\leq_l$, respectively. Clearly, $(L_d, \leq_c)$ is a finite lattice with the componentwise ordering and $(L_d, \leq_l)$ is a finite lattice with the lexicographic ordering.

Let f_c and f_l be an order preserving mapping from L_d into itself under the componentwise ordering and the lexicographic ordering, respectively.

1.1.1 Oracle Function Model

When $f_l(\cdot)$ and $f_c(\cdot)$ are given as oracle functions, we develop a complete understanding for finding a Tarski's fixed point as well as determining the uniqueness of Tarski's fixed point in both the lexicographic ordering and the componentwise ordering lattices.

We develop an algorithm of time complexity $O((\log |L|)^d)$ to find a Tarski's fixed point on the componentwise ordering lattice $(L, \leq_c)$, for any dimension d. This algorithm is based on the binary search method. We first present the algorithm when $d = 2$. Following a similar principle, this algorithm can be generalized to any constant dimension. This is the first known polynomial time algorithm for finding a Tarski's fixed point in terms of the componentwise ordering. In the literatures, a polynomial time algorithm was only known for the total order lattices (Chang et al.) [8], where any pair of lattice points are comparable.

On the other hand, given a general lattice $(L, \preceq)$ with one already known fixed point, to find out whether it is unique will take $\Omega(|L|)$ time for any algorithm. For a componentwise ordering lattice, we derive a $\Theta(N_1 + N_2 + \cdots + N_d)$ matching bound for determining the uniqueness of Tarski's fixed point, where $L = \{\mathbf{x} \in Z^d \mid \mathbf{a} \leq \mathbf{x} \leq \mathbf{b}\}$ and $N_i = b_i - a_i$. In addition, we prove this matching bound for both deterministic algorithm and randomized algorithm.

For a lexicographic ordering lattice, it can be viewed as a componentwise ordering lattice with dimension one by an appropriate polynomial time transformation to change the oracle function for the d-dimension space to an oracle function on the 1-dimension space. All the above results can be transplanted onto the lexicographic ordering lattice with a set of related parameters.

1.1.2 Polynomial Function Model

Under the polynomial time function model, our polynomial time algorithm applies when the dimension is any finite constant. When the dimension is used as a part of the input size in unary, we first present a polynomial-time reduction of integer programming to an order preserving mapping f from a componentwise ordering lattice L into itself. As a result of this reduction, we obtain that, given f as a polynomial time function, determining whether f has a unique fixed point in L is a Co-NP hard problem. Furthermore, even when the dimension is one, we also find a polynomial-time reduction of determining the feasibility of an integer programming to the uniqueness of Tarski's fixed point in a lexicographic lattice. This shows that determining the uniqueness of Tarski's fixed point in a lexicographic lattice is Co-NP hard though there exists a polynomial-time algorithm for computing a Tarski's fixed point in a lexicographic lattice in any dimension.

1.2 Related Work

For the oracle function model, when the lattice $(L, \preceq)$ has a total order, i.e., all the point in the lattice is comparable, there is a matching bound of $\theta(\log |L|)$ in Chang et al.[8]. It is a special case of

our result for the oracle function model since a total order lattice is equivalent to a componentwise ordering lattice of dimension one.

1.3 Organization

The rest of the paper is organized as follows. First, in Section 2, we present definitions as well as the difference of the polynomial function model and the oracle function model. We develop polynomial time algorithms in oracle function model for componentwise ordering and lexicographic ordering in Section 3. In Section 4, we derive the matching bound for determining the uniqueness of Tarski's fixed point under the oracle function model. We prove co-NP hardness for determining the uniqueness of Tarski's fixed point under the polynomial function model in Section 5. We conclude with discussion and remarks on our results and open problems in Section 6.

2 Preliminaries

In this section, we first introduce formal definitions of some relevant concepts as well as Tarski's fixed point theorem. We then compare the difference between the oracle function model and the polynomial function model.

2.1 Definitions and Theorems

Definition 1. *(Partial Order vs. Total Order) A relationship $\preceq$ on a set L is a partial order if it satisfies reflexivity ($\forall \mathbf{a} \in L : \mathbf{a} \preceq \mathbf{a}$); antisymmetry ($\mathbf{a} \preceq \mathbf{b}$ and $\mathbf{b} \preceq \mathbf{a}$ implies $\mathbf{a} = \mathbf{b}$); transitivity ($\mathbf{a} \preceq \mathbf{b}$ and $\mathbf{b} \preceq \mathbf{c}$ implies $\mathbf{a} \preceq \mathbf{c}$). It is a total order if $\forall \mathbf{a}, \mathbf{b} \in L$: either $\mathbf{a} \preceq \mathbf{b}$ or $\mathbf{b} \preceq \mathbf{a}$.*

Definition 2. *(Lattice) $(L, \preceq)$ is a lattice if*

1. *L is a partial ordered set;*
2. *There are two operations: meet $\wedge$ and join $\vee$ on any pair of elements $\mathbf{a}, \mathbf{b}$ of L such that $\mathbf{a}, \mathbf{b} \preceq \mathbf{a} \vee \mathbf{b}$ and $\mathbf{a} \wedge \mathbf{b} \preceq \mathbf{a}, \mathbf{b}$*

The lattice is complete if, for any subset $A = \{\mathbf{a}_1, \mathbf{a}_2, \cdots, \mathbf{a}_k\} \subseteq L$, there is a unique meet and a unique join: $\bigwedge A = (\mathbf{a}_1 \wedge \mathbf{a}_2 \wedge \cdots \wedge \mathbf{a}_k)$ and $\bigvee A = (\mathbf{a}_1 \vee \mathbf{a}_2 \vee \cdots \vee \mathbf{a}_k)$.

For simplicity, we use L for a lattice when no ambiguity exists on $\preceq$. We should specify $\preceq$ whenever it is necessary.

Definition 3. *(Order Preserving Function) A function f on a lattice $(L, \preceq)$ is order preserving if $\mathbf{a} \preceq \mathbf{b}$ implies $f(\mathbf{a}) \preceq f(\mathbf{b})$.*

Theorem 1. *(Tarski's Fixed Point Theorem)[23]. If L is a complete lattice and f an increasing from L to itself, there exists some $\mathbf{x}^* \in L$ such that $f(\mathbf{x}^*) = \mathbf{x}^*$, which is a fixed point of f.*

This theorem guarantees the existence of fixed points of any order-preserving function $f : L \to L$ on any nonempty complete lattice.

Definition 4. *(Lexicographic Ordering Function). Given a set of points on a d-dimensional space R^d, the lexicographic ordering function $\leq_l$ is defined as:*
$\forall \mathbf{x}, \mathbf{y} \in R^d$, $\mathbf{x} \leq_l \mathbf{y}$ if either $\mathbf{x} = \mathbf{y}$ or $x_i = y_i$, $i = 1, 2, \ldots, k-1$, and $x_k < y_k$ for some $k \leq d$.

Definition 5. *(Componentwise Ordering Function). Given a set of points on a d-dimensional space, the componentwise ordering function $\leq_c$ is defined as:*

$\forall \mathbf{x}, \mathbf{y} \in R^d$, $\mathbf{x} \leq_c \mathbf{y}$ *if* $\forall i \in \{1, 2, \cdots, d\} : x_i \leq y_i$.

2.2 The oracle function model versus the polynomial time function model.

In recent years, the computational complexities of the fixed point computation has received intensive examinations, under both the oracle function model and the polynomial time function model.

The differences of these two models have an impact on the computational complexities of them. For the oracle function model, the function value $f(x)$ is revealed only if the value x is specified to the oracle. The oracle is restricted to be consistent in that the answer has to be consistent in the returned values, i.e., the returned values of the same input x in different queries must be the same. However, it still has the flexibility in the choices of function values dependent on the algorithm designed to find the guaranteed fixed point. In this sense, the oracle has a huge range of functions to choose from, and hence makes it difficult for an algorithm to deal with all the answer sequences with respect to its query sequences. The oracle model allows for any function to be used and therefore often demands a large complexity to handle every case. The polynomial time function model, on the other hand, provides, a prior, a function computable in polynomial time with respect to the input size, such as a circuit of AND/OR/NOT gates of size polynomial in the input size. It is possible that the algorithmic designer could utilize the knowledge of the polynomial time computable function to design a fast fixed point computation procedure though there has no universal framework to tell us how to do it. Therefore, the polynomial time function model can only accommodate a much less number of functions than the oracle function model. Hence, one may only need to consider a smaller set of functions than that in the oracle function model. Hence, the oracle function model often demands a large computational complexity. More details on comparing these two models in the general fixed point computation can be found in (Deng et al.)[13].

3 Polynomial Time Algorithm under Oracle Function Model

In this section, we consider the complexity of finding a Tarski's fixed point in any constant dimension d with the function value f given by an oracle. Chang et al. [8] proved that a fixed point can be found in time polynomial when the given lattice is total order.

Define $L = \{\mathbf{x} \in Z^d \mid \mathbf{a} \leq \mathbf{x} \leq \mathbf{b}\}$, where $\mathbf{a}$ and $\mathbf{b}$ are two finite vectors of Z^d with $\mathbf{a} < \mathbf{b}$.

Theorem 2. *(Chang et al.)[8] When $(L, \preceq)$ is given as input and the order preserving function f is given as oracles, a Tarski's fixed point can be found in time $O(\log |L|)$ on a finite lattice when $\preceq$ is a total order on L.*

Since any two vectors in the lexicographic ordering is comparable, the lexicographic ordering is a total order. We have

Corollary 1. *When $(L, \preceq)$ is given as input and the order preserving function f is given as oracles, a Tarski's fixed point can be found in time $O(\log |L|)$ on a finite lattice when $\preceq$ is the lexicographic ordering on L.*

The proof is rather standard utilizing the total order property of the lexicographic ordering. As the componentwise ordering lattice cannot be modeled as a total order, it leaves open the oracle

complexity of finding a fixed point in componentwise ordering lattice. Here we show that this problem is also polynomial time solvable, by designing a polynomial algorithm to find a fixed point of f in time $O((\log |L|)^d)$ given componentwise ordering lattice L.

The algorithm exploits the order properties of the componentwise lattice and applying the binary search method with a dimension reduction technique. To illustrate the main ideas, we first consider the 2D case before moving on to the general case.

Without loss of generality (WLOG), we assume L is an $N \times N$ square centered at point $(0, 0)$. The componentwise ordering is denoted as $\leq_c$.

Algorithm 3.1. *Point_check() (A polynomial algorithm for 2D lattice)*

- *Input:*

  2-dimensional lattice $(L, \leq_c)$, $|L| = N^2$ *(Input size to the oracle is* $2 \log N$ *since the input size for both dimensions to the oracle is* $\log N$*.)*

  Oracle function f*.* f *is an order preserving function.* $\forall \mathbf{x} \in L, f(\mathbf{x}) \in L$ *and* $f(\mathbf{x}) \leq_c f(\mathbf{y})$ *if* $\mathbf{x} \leq_c \mathbf{y}, \forall \mathbf{x}, \mathbf{y} \in L$

- *Point_check(*L, f*)*

  Let $\mathbf{x}^0$ *be the center point in* L*. Let* $\mathbf{x}^L$ *be the left most point in* L *such that* $\mathbf{x}^L_2 = \mathbf{x}^0_2$*. Let* $\mathbf{x}^R$ *be the right most point in* L *such that* $\mathbf{x}^R_2 = \mathbf{x}^0_2$*.*

  1. *If* $f(\mathbf{x}^0) = \mathbf{x}^0$*,return(*$\mathbf{x}^0$*);end;*
  2. *If* $f(\mathbf{x}^0) \geq_c \mathbf{x}^0$*,*$L' = \{\mathbf{x} | \mathbf{x} \geq_c \mathbf{x}^0, \mathbf{x} \in L\}$*. Point_check(*$L', f$*);*
  3. *If* $f(\mathbf{x}^0) \leq_c \mathbf{x}^0$*,* $L'' = \{\mathbf{x} | \mathbf{x} \leq_c \mathbf{x}^0, \mathbf{x} \in L\}$*. Point_check(*$L'', f$*);*
  4. *If* $f(\mathbf{x}^0)_1 < x^0_1$ *and* $f(\mathbf{x}^0)_2 > x^0_2$*, Binary_Search(*$\mathbf{x}^L, \mathbf{x}^0$*);*
  5. *If* $f(\mathbf{x}^0)_1 > x^0_1$ *and* $f(\mathbf{x}^0)_2 < x^0_2$*, Binary_Search(*$\mathbf{x}^0, \mathbf{x}^R$*);*

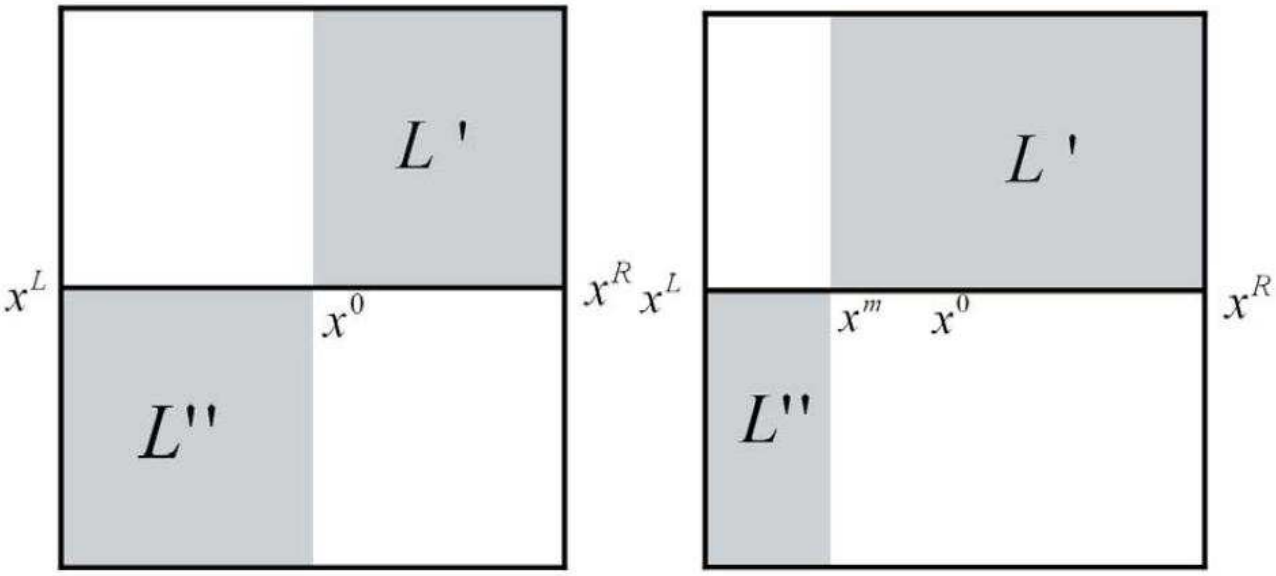
- *Binary_Search(*$\mathbf{x}, \mathbf{y}$*)*

  Let $\mathbf{x}^m = \lfloor 1/2(\mathbf{x} + \mathbf{y}) \rfloor$

  1. *If* $f(\mathbf{x}^m) = \mathbf{x}^m$*,return(*$\mathbf{x}^m$*);end;*
  2. *If* $f(\mathbf{x}^m) \geq_c \mathbf{x}^m$*,* $L' = \{\mathbf{x} | \mathbf{x} \geq_c \mathbf{x}^m, \mathbf{x} \in L\}$*. Point_check(*$L', f$*);*
  3. *If* $f(\mathbf{x}^m) \leq_c \mathbf{x}^m$*,* $L'' = \{\mathbf{x} | \mathbf{x} \leq_c \mathbf{x}^m, \mathbf{x} \in L\}$*. Point_check(*$L'', f$*);*
  4. *If* $f(\mathbf{x}^m)_1 < x^m_1$ *and* $f(\mathbf{x}^m)_2 > x^m_2$*, Binary_Search(*$\mathbf{x}, \mathbf{x}^m$*);*
  5. *If* $f(\mathbf{x}^m)_1 > x^m_1$ *and* $f(\mathbf{x}^m)_2 < x^m_2$*, Binary_Search(*$\mathbf{x}^m, \mathbf{y}$*);*

Theorem 3. *When the order preserving function* f *is given as an oracle, a Tarski's fixed point can be found in time* $O(\log^2 N)$ *on a finite 2D lattice formed by integer points of a box with side length* N *by using Algorithm 3.1 Point_check.*

Proof. Start from a lattice of size $|L|$, we first prove that in at most $O(\log N)$ steps the above algorithm either finds the fixed point or reduces the input lattice to size $|L|/2$.



(a) Point_check() (b) Binary_Search()

Figure 1: A polynomial algorithm for 2D Lattice

Consider the algorithm Point_check(L, f).

1. Case I: If $f(\mathbf{x}^0) = \mathbf{x}^0$, $\mathbf{x}^0$ is the fixed point which is found in 1 step.

2. Case II: If $f(\mathbf{x}^0) \geq_c \mathbf{x}^0$, since f is an order preserving function, $\forall \mathbf{y} \geq_c \mathbf{x}^0$, we have $f(\mathbf{y}) \geq_c f(\mathbf{x}^0) \geq_c \mathbf{x}^0$. Let $L' = \{\mathbf{x} | \mathbf{x} \geq_c \mathbf{x}^0, \mathbf{x} \in L\}$. Define $f'(\mathbf{x}) = f(\mathbf{x}), \forall \mathbf{x} \in L'$. Then $f' : L' \to L'$ is a order preserving function on the complete lattice L'. By Tarski's fixed point theorem, there must exist a fixed point in L'. Next we only need to check L' which is only 1/4 size of $|L|$.

3. Case III: If $f(\mathbf{x}^0) \leq_c \mathbf{x}^0$, similar to the analysis in Case II, we only need to consider $L'' = \{\mathbf{x} | \mathbf{x} \leq_c \mathbf{x}^0, \mathbf{x} \in L\}$ which is only 1/4 size of $|L|$ in the next step.

4. Case IV: If $f(\mathbf{x}^0)_1 < x_1^0$ and $f(\mathbf{x}^0)_2 > x_2^0$, we prove that Binary_Search$(\mathbf{x}^L, \mathbf{x}^0)$ finds a fixed point or reduce the size of the lattice by half in $\log \frac{N}{2}$ steps. Since f is an order preserving function, $\forall$ adjacent points $\mathbf{u} \leq_c \mathbf{v} \in L$ (i.e., $||u - v||_\infty = 1$), it is impossible that $f(\mathbf{u})_1 > u_1$ and $f(\mathbf{v})_1 < v_1$. Thus, on a line segment $[\mathbf{x}, \mathbf{y}]$ where $x_2 = y_2$, if $f(\mathbf{x})_1 \geq x_1$ and $f(\mathbf{y})_1 < y_1$, there must exist a point $\mathbf{z}$ such that $f(\mathbf{z})_1 = z_1$. On the other hand, we have $f(\mathbf{x}^0)_1 < x_1^0$ and by the boundary condition $f(\mathbf{x}^L)_1 \geq x_1^L$, therefore, there must exist a point $\mathbf{x}' \in [\mathbf{x}^L, x^0)$ such that $f(\mathbf{x}')_1 = x'_1$. This point $\mathbf{x}'$ can be found in time $\log \frac{N}{2}$ by using binary search. If $f(\mathbf{x}')_2 > x'_2$, similar to the analysis in Case II, we only need to consider $L' = \{\mathbf{x} | \mathbf{x} \geq_c \mathbf{x}', \mathbf{x} \in L\}$ which is at most 1/2 size of $|L|$ in the next step. If $f(\mathbf{x}')_2 < x'_2$, we only need to consider $L'' = \{\mathbf{x} | \mathbf{x} \leq_c \mathbf{x}', \mathbf{x} \in L\}$ which is at most 1/4 size of $|L|$ in the next step. If $f(\mathbf{x}')_2 = x'_2$, then $\mathbf{x}'$ is the fixed point.

5. Case V: If $f(\mathbf{x}^0)_1 > x_1^0$ and $f(\mathbf{x}^0)_2 < x_2^0$, similarly, we can prove that Binary_Search$(\mathbf{x}^0, \mathbf{x}^R)$finds a fixed point or reduce the size of the lattice by half in $\log \frac{N}{2}$ steps.

The size of the lattice is reduced by half in every $O(\log N)$ steps. Therefore, the algorithm finds a fixed point in at most $O(\log N \times \log L) = O(\log^2 N)$ steps. □

The above algorithm can be generalized to any constant dimensional lattice with $L = \{\mathbf{x} \in Z^d \mid \mathbf{a} \leq \mathbf{x} \leq \mathbf{b}\}$, where $\mathbf{a}$ and $\mathbf{b}$ are two finite vectors of Z^d with $\mathbf{a} < \mathbf{b}$. We reduce a $(d+1)$-dimension problem to a d-dimension one. Assume we have an algorithm for a d-dimensional problem with time complexity $O(\log^d |L|)$. Let the algorithm be $A_d(L, f)$.

Consider a $d+1$-dimensional lattice $(L, \leq_c)$. Choose the central point in L, and denote it by $\mathbf{O} = (O_1, O_2, \cdots, O_{d+1})^T$. Take the section of L by a hyperplane parallel to $x_{d+1} = 0$ passing through $\mathbf{O}$. Denote it as L^d. Clearly, it is a d-dimensional lattice. We define a new oracle function f_d on L^d, based on the oracle function f on L. Define $f_d(x_1, x_2, \cdots, x_d) = (y_1, y_2, \cdots, y_d)$, if $f(x_1, x_2, \cdots, x_d, O_{d+1}) = (y_1, y_2, \cdots, y_d, y_{d+1})$. We apply the algorithm $A_d(L^d, f_d)$ to obtain a Tarski's fixed point in time $(\log |L|)^d$. Let the fixed point be denoted by $\mathbf{x}^*$. Therefore, $f(\mathbf{x}^*) = (\mathbf{x}^*, O_{d+1}) + a\mathbf{e}_{d+1}$ or $f(\mathbf{x}^*) = (\mathbf{x}^*, O_{d+1}) - a\mathbf{e}_{d+1}$, where a is some constant, $\mathbf{e}_{d+1}$ is a $d+1$ dimensional unit vector with 1 on its $d+1$th position.

In either case, we obtain a new box $\mathbf{B}$ with size no more than half of the original box defined by $[\mathbf{a}, \mathbf{b}]$, such that $f(\cdot)$ maps all points in $\mathbf{B}$ into $\mathbf{B}$ and is order preserving. We can apply the algorithm recursively on $\mathbf{B}$. The base case can be handled easily. Therefore the total time is

$$T(|L|^{d+1}) \leq T(\frac{|L|^{d+1}}{2}) + O(\log^d |L|).$$

It follows that $T(|L|^{d+1}) = O(\log^d |L|)$.

Algorithm 3.2. *Fixed_point() (A polynomial algorithm for any constant dimensional lattice)*

- *Input:*

  d dimensional lattice L^d, WLOG, $|L^d| = N^d$ (Input size to the oracle is $d \log N$ since the input size for both dimensions to the oracle is $\log N$.)

  Oracle function f^d. f^d is an order preserving function. $\forall \mathbf{x} \in L^d, f^d(\mathbf{x}) \in L^d$ and $f^d(\mathbf{x}) \leq_c f^d(\mathbf{y})$ if $\mathbf{x} \leq_c \mathbf{y}, \forall \mathbf{x}, \mathbf{y} \in L^d$

- *Fixed_point(L^d)*

  1. *If $d > 1$*
     (a) *Let $\mathbf{x}^0$ be the center point in L^d.*
     (b) *Let $L^{d-1} = \{\mathbf{x} = (x_1, x_2, \cdots, x_{d-1}) | (\mathbf{x}, x_d^0) \in L^d\}$.*
     (c) *Let $f^{d-1}(\mathbf{x}) = (f^d(\mathbf{x}, x_d^0)_1, f^d(\mathbf{x}, x_d^0)_2, \cdots, f^d(\mathbf{x}, x_d^0)_{d-1})$.*
     (d) *$\mathbf{x}^*$ =Fixed_point(L^{d-1}).*
     (e) *If $f^d(\mathbf{x}^*, x_d^0)_d > x_d^0$, $L^d = \{\mathbf{x} | \mathbf{x} \geq (\mathbf{x}^*, x_d^0)\}$; Fixed_point($L^d$);*
     (f) *If $f^d(\mathbf{x}^*, x_d^0)_d < x_d^0$, $L^d = \{\mathbf{x} | \mathbf{x} \leq (\mathbf{x}^*, x_d^0)\}$; Fixed_point($L^d$);*
     (g) *If $f^d(\mathbf{x}^*, x_d^0)_d = x_d^0$, return $(\mathbf{x}^*, x_d^0)$; end;*
  2. *If $d = 1$, let $\mathbf{x}^L$ be the left end point and $\mathbf{x}^R$ be the right end point. binary_search($\mathbf{x}^L, \mathbf{x}^R, f^d$).*

- *binary_search($\mathbf{x}, \mathbf{y}, f$)*

1. *If* $f(\mathbf{x}^L) = 0$, *output* $\mathbf{x}^L$;
2. *else if* $f(\mathbf{x}^R) = 0$, *output* $\mathbf{x}^R$;
3. *else*
   (a) *If* $f(\lfloor 1/2(\mathbf{x}^L + \mathbf{x}^R)\rfloor) < \lfloor 1/2(\mathbf{x}^L + \mathbf{x}^R)\rfloor$, *binary_search(*$\mathbf{x}^L, \lfloor 1/2(\mathbf{x}^L + \mathbf{x}^R)\rfloor, f$*);*
   (b) *If* $f(\lfloor 1/2(\mathbf{x}^L + \mathbf{x}^R)\rfloor) > \lfloor 1/2(\mathbf{x}^L + \mathbf{x}^R)\rfloor$, *binary_search(*$\lfloor 1/2(\mathbf{x}^L + \mathbf{x}^R)\rfloor, \mathbf{x}^R, f$*);*
   (c) *else output* $\mathbf{x}^*$.

Theorem 4. *When the order preserving function f is given as an oracle, a Tarski's fixed point can be found in time $O(\log^d |L|)$ on a componentwise ordering lattice $(L, \leq_c)$.*

4 Determining the Uniqueness under Oracle Function Model

It has been a natural question to check whether there is another fixed point after finding the first one, such as in the applications for finding a Nash equilibrium (Echenique)[14] . In this section we develop a lower bound that, given a general lattice L with one already known fixed point, determining whether it is unique will take $\Omega(|L|)$ time for any algorithm. Even for the componentwise ordering lattice, we also derive a $\Theta(N_1 + N_2 + \cdots + N_d)$ matching bound for determining the uniqueness of fixed points even for randomized algorithms. The technique builds on and further reveals crucial properties of mathematical structures for fixed points.

Theorem 5. *Given a lattice $(L, \preceq)$, an order preserving function f and a fixed point $\mathbf{x}^0$, it takes time $\Omega(|L|)$ for any deterministic algorithm to decide whether there is a unique fixed point.*

Proof. Consider the lattice on a real line: $0 \prec 1 \prec 2 \prec \cdots \prec L-1$. Let $x^0 = 0$, define $f(0) = 0$ and $f(x) = x - 1$ for all $x \geq 1$ except a possible fixed point x^*. $f(x^*) = x^*$ or $f(x^*) = x^* - 1$ which is not known until we query x^*. Given a deterministic algorithm A, define y_j be the j-th item A queried in its effort to find x^*. Our adversary will answer $x - 1$ whenever A asks for $f(x)$ until the last item when the adversary answers x. Clearly this derives a lower bound of L. □

For a randomized algorithm R, let p_{ij} be the probability R queries $x = i$ on its j-th query. Let k be the total number of queries R makes. We have:

$$\sum_{j=1}^{k} \sum_{i=0}^{|L|-1} p_{ij} = k.$$

Therefore, there exists i^* such that

$$\sum_{j=1}^{k} p_{i^* j} \leq \frac{k}{|L|}.$$

The adversary will place $f(i^*) = i^*$, which is queried with probability $\frac{k}{|L|} < 1/2$ when we choose $k = \frac{|L|-1}{2}$. Therefore, we have

Theorem 6. *Given a lattice $(L, \preceq)$, an order preserving function f and a fixed point $\mathbf{x}^0$, it takes time $\Omega(|L|)$ for any randomized algorithm to decide whether there is a unique fixed point with probability at least $1/2$.*

As we noted before, for a lexicographic ordering lattice, it can be viewed as a total ordering lattice or componentwise ordering lattice with dimension one by an appropriate polynomial time transformation to change the oracle function for the d-dimension space to an oracle function on the 1-dimension space. Therefore,

Corollary 2. *Given a lattice $(L, \leq_l)$, an order preserving function f and a fixed point $\mathbf{x}^0$, it takes time $\Omega(|L|)$ both for any deterministic algorithm and for any randomized algorithm to decide whether there is a unique fixed point with probability at least $1/2$.*

Next we consider a componentwise ordering lattice.

Theorem 7. *Given the componentwise ordering lattice $L = N_1 \times N_2 \times \cdots \times N_d$ of d dimensions and an order preserving function f as well as a known fixed point $\mathbf{x}^0$. The deterministic oracle complexity is $\theta(N_1 + N_2 + \cdots + N_d)$ to decide whether there is a unique fixed point.*

Proof. For dimension $d \geq 2$, let $L = \{\mathbf{x} \in Z : \mathbf{0} \leq_c \mathbf{x} \leq_c (N_1, N_2, \cdots, N_d)\}$. For $\mathbf{x} = (x_1, x_2, \cdots, x_d)$, let $maxindex(\mathbf{x}) = \max\{i : x_i > 0\}$ for any non-zero vector $\mathbf{x}$. Define $auxi(\mathbf{x}) = -\mathbf{e}_{maxindex(\mathbf{x})}$ where $\mathbf{e}_i$ is a unit vector in the i-th coordinate. Therefore, $auxi(\cdot)$ is well defined on nonzero vectors in the lattice L. One example of two-dimensional case is demonstrated in Fig. 2. The fixed point is denoted by the red color. The direction of all the other points are defined by the function $auxi(\cdot)$.

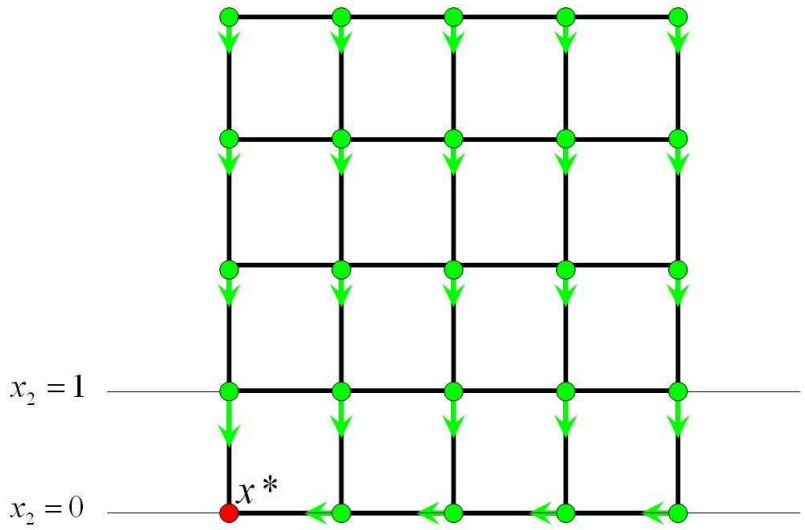


Figure 2: $auxi(\mathbf{x})$

The adversary will set $g(\mathbf{x}) = f(\mathbf{x}) - \mathbf{x}$ to be $auxi(\mathbf{x})$ except at certain points (to be decided according to the algorithm) where it may hide a zero point.

1. Proof of the Lower Bound:

First consider $\mathbf{x}$ such that $x_d = 0$. It constitutes a solution of $d-1$ dimensions. By inductive hypothesis, it requires time $N_1 + N_2 + \cdots + N_{d-1}$ to decide whether or not there is one zero point at $x_d = 0$.

Second, when there is no such zero point, we need to decide if there is a zero point at $\mathbf{x}$ with $x_d > 0$. Fixing any $i > 0$, we will set, for all $\mathbf{x}$ with $x_d = i$, $g(\mathbf{x}) = 0$ whenever none of such $\mathbf{x}$ is queried, and set $g(\mathbf{x}) = -e_i$ otherwise. This will take N_d queries.

One may note that the adversary always answers a non-zero value. In fact, for any pair $i = maxindex(\mathbf{x})$ and $j = x_i$ not queried, the adversary can make $g(\mathbf{x}) = 0$ without violating the order preserving property.

2. Proof of the Upper Bound:

   We design an algorithm which always queries the componentwise maximum point of the lattice $\mathbf{x}^{\max} = (N_1, N_2, \cdots, N_d)$. We should have $g(\mathbf{x}^{\max}) \leq_c \mathbf{0}$. We are done if it is zero. Otherwise, there must exist some i, such that $g(\mathbf{x}^{\max})_i < 0$. The problem is reduced to a smaller lattice $L' = \{\mathbf{x} \in Z : \mathbf{0} \leq_c \mathbf{x} \leq_c (N_1, N_2, \cdots, N_{i-1}, N_i - 1, N_{i+1}, \cdots, N_d)\}$ which has a total sum of side lengths at most $N_1 + N_2 + \cdots + N_d - 1$. The claim follows.

□

For the randomized lower bound, it follows in the same way as in the one-dimensional case for a general lattice. We can always set $f(\mathbf{x}) = 0$ for all $\mathbf{x}$ with $i = maxindex(\mathbf{x})$ and $j = x_i$ if none of such $\mathbf{x}$ is queried.

Corollary 3. *Given the componentwise ordering lattice $L = N_1 \times N_2 \times \cdots \times N_d$ of d dimensions and an order preserving function f as well as a known fixed point $\mathbf{x}^0$. It takes time $\theta(N_1 + N_2 + \cdots + N_d)$ for any randomized algorithm to decide whether there is a unique fixed point with probability at least $1/2$.*

5 Determining Uniqueness under Polynomial Function Model

In this section, we consider the dimension as a part of the input size in unary and develop a hardness proof for the polynomial function model for determining the uniqueness of a given fixed point. We start with a polynomial-time reduction from a special class of integer programming which is NP-complete to one of finding a second Tarski's fixed point, by deriving an order preserving mapping f from a componentwise ordering lattice L into itself, with a given fixed point. Therefore, given f as a polynomial time function with a known fixed point, determining whether f has another fixed point in L is an NP-hard problem. In other words, determining the uniqueness of a Tarski's fixed point is co-NP-hard.

Furthermore, even for the case when the dimension is one, the uniqueness problem is still co-NP-hard. This can be done by designing a polynomial-time reduction of determining the feasibility of an integer programming to the uniqueness of Tarski's fixed point in a lexicographic lattice. As the lexicographic order defines a total order, it can be reduced to a one dimensional problem by finding a polynomial time algorithm for the order function calculation. It then follows that determining the uniqueness of Tarski's fixed point in a lexicographic lattice is Co-NP hard though there exists

a polynomial-time algorithm for finding one Tarski's fixed point in a lexicographic lattice in any dimension.

Let $P = \{\mathbf{x} \in R^n \mid A\mathbf{x} \leq \mathbf{b}\}$ be a full-dimensional polytope, where A is an $m \times n$ rational matrix satisfying that each row of A has at most one positive entry and $\mathbf{b}$ a rational vector of R^m. It has been shown in (Lagarias)[18] that

Theorem 8. *[18] Determining whether there is an integer point in P is an NP-complete problem.*

5.1 Co-NP-hard in Componentwise Ordering

Let $N = \{1, 2, \ldots, n\}$. For any real number α, let $\lfloor \alpha \rfloor$ denote the greatest integer less than or equal to α and $\lceil \alpha \rceil$ the smallest integer greater than or equal to α. For any vector $\mathbf{x} = (x_1, x_2, \ldots, x_n)^\top \in R^n$, let $\lfloor \mathbf{x} \rfloor = (\lfloor x_1 \rfloor, \lfloor x_2 \rfloor, \ldots, \lfloor x_n \rfloor)^\top$ and $\lceil \mathbf{x} \rceil = (\lceil x_1 \rceil, \lceil x_2 \rceil, \ldots, \lceil x_n \rceil)^\top$. Given these notations, we present a polynomial-time reduction of integer programming, which is as follows.

For any $\mathbf{x} \in R^n$, let $P(\mathbf{x}) = \{\mathbf{y} \in P \mid \mathbf{y} \leq_c \mathbf{x}\}$. Then, as a direct result of the property of the matrix A, one can easily obtain that

Lemma 1. *For any given $\mathbf{x} \in R^n$, if $\mathbf{x}^1 = (x_1^1, x_2^1, \ldots, x_n^1)^\top \in P(\mathbf{x})$ and $\mathbf{x}^2 = (x_1^2, x_2^2, \ldots, x_n^2)^\top \in P(\mathbf{x})$, then*

$$\bar{\mathbf{x}} = \max(\mathbf{x}^1, \mathbf{x}^2) = (\max\{x_1^1, x_1^2\}, \max\{x_2^1, x_2^2\}, \ldots, \max\{x_n^1, x_n^2\})^\top \in P(\mathbf{x}).$$

Let $\mathbf{e} = (1, 1, \ldots, 1)^\top \in R^n$. For any given $\mathbf{v} \in R^n$, if $P(\mathbf{v}) \neq \emptyset$, Lemma 1 implies that $\max_{\mathbf{x} \in P(\mathbf{v})} \mathbf{e}^\top \mathbf{x}$ has a unique solution, which we denote by $\mathbf{x}^{\mathbf{v}} = (x_1^{\mathbf{v}}, x_2^{\mathbf{v}}, \ldots, x_n^{\mathbf{v}})^\top$.

Lemma 2. $\mathbf{x} \leq_c \mathbf{x}^{\mathbf{v}}$ *for all* $\mathbf{x} \in P(\mathbf{v})$.

Proof. Suppose that there is a point $\mathbf{x}^0 = (x_1^0, x_2^0, \ldots, x_n^0)^\top \in P(\mathbf{v})$ with $x_k^0 > x_k^{\mathbf{v}}$ for some $k \in N$. Then, Lemma 1 implies that

$$\mathbf{x}^{\mathbf{v}0} = (\max\{x_1^0, x_1^{\mathbf{v}}\}, \max\{x_2^0, x_2^{\mathbf{v}}\}, \ldots, \max\{x_n^0, x_n^{\mathbf{v}}\})^\top \in P(\mathbf{v}).$$

Thus, $e^\top \mathbf{x}^{0\mathbf{v}} > e^\top \mathbf{x}^{\mathbf{v}} = \max_{\mathbf{x} \in P(\mathbf{v})} \mathbf{e}^\top \mathbf{x}$. A contradiction arises. This completes the proof. □

Let $\mathbf{x}^{\max} = (x_1^{\max}, x_2^{\max}, \ldots, x_n^{\max})^\top$ be the unique solution of $\max_{\mathbf{x} \in P} \mathbf{e}^\top \mathbf{x}$ and $\mathbf{x}^{\min} = (x_1^{\min}, x_2^{\min}, \ldots, x_n^{\min})^\top$ with $x_j^{\min} = \min_{\mathbf{x} \in P} x_j$, $j = 1, 2, \ldots, n$. Then, $\mathbf{x}^{\min} \leq_c \mathbf{x} \leq_c \mathbf{x}^{\max}$ for all $\mathbf{x} \in P$. Let

$$D(P) = \{\mathbf{x} \in Z^n \mid \mathbf{x}^l \leq_c \mathbf{x} \leq_c \mathbf{x}^u\},$$

where $\mathbf{x}^u = \lfloor \mathbf{x}^{\max} \rfloor$ and $\mathbf{x}^l = \lfloor \mathbf{x}^{\min} \rfloor$. Thus, $D(P)$ contains all integer points in P. Without loss of generality, we assume that $\mathbf{x}^l <_c \mathbf{x}^{\min}$ (Let $x_i^l = x_i^{\min} - 1$ if $x_i^l = x_i^{\min}$ for some $i \in N$). Obviously, the sizes of both $\mathbf{x}^l$ and $\mathbf{x}^u$ are bounded by polynomials of the sizes of the matrix A and the vector $\mathbf{b}$ since $\mathbf{x}^l$ and $\mathbf{x}^u$ are obtained from the solutions of linear programs with rational data.

For $\mathbf{x} \in R^n$, we define $h(\mathbf{x}) = \lfloor d(\mathbf{x}) \rfloor$ with

$$d(\mathbf{x}) = \begin{cases} \mathbf{x}^l & \text{if } P(\mathbf{x}) = \emptyset, \\ \text{argmax}_{\mathbf{y} \in P(\mathbf{x})} \mathbf{e}^\top \mathbf{y} & \text{otherwise.} \end{cases}$$

It follows from Lemma 2 that $d(\mathbf{x})$ is well defined.

Example 1. *Consider* $P = \{\mathbf{x} \in R^3 \mid A\mathbf{x} \leq_c \mathbf{b}\}$, *where*

$$A = \begin{pmatrix} 2 & -1 & 0 \\ -1 & 3 & 0 \\ 0 & 0 & 2 \\ 0 & -1 & -1 \end{pmatrix}$$

and $\mathbf{b} = (0, -10, 10, 0)^\top$. *For* $\mathbf{y} = (-3, -4, 5)^\top$, $h(\mathbf{y}) = (-3, -5, 5)^\top$. *An illustration of* h *can be found in Fig.3.*

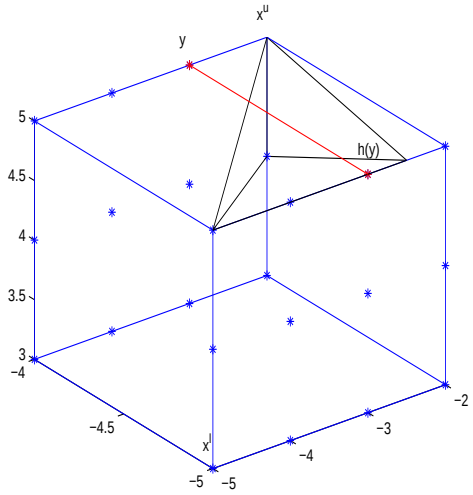


Figure 3: An Illustration of h

Lemma 3. *h is an order preserving mapping from R^n to $D(P)$. Moreover, $h(\mathbf{x}^*) = \mathbf{x}^* \neq \mathbf{x}^l$ if and only if $\mathbf{x}^*$ is an integer point in P.*

Proof. Let $\mathbf{x}^1$ and $\mathbf{x}^2$ be two different points of R^n with $\mathbf{x}^1 \leq_c \mathbf{x}^2$. Then, $P(\mathbf{x}^1) \subseteq P(\mathbf{x}^2)$. Thus, from the definition of $d(\mathbf{x})$, we obtain that $\mathbf{x}^{\min} \leq_c d(\mathbf{x}^1) \leq_c d(\mathbf{x}^2) \leq_c \mathbf{x}^{\max}$. The first part of the lemma follows immediately.

Let $\mathbf{x}^*$ be an integer point in P. Then,

$$d(\mathbf{x}^*) = \text{argmax}_{\mathbf{y} \in P(\mathbf{x}^*)} \mathbf{e}^\top \mathbf{y} = \mathbf{x}^*.$$

Thus, $h(\mathbf{x}^*) = \mathbf{x}^* \neq \mathbf{x}^l$.

Let $\mathbf{x}^*$ be a point in R^n satisfying that $h(\mathbf{x}^*) = \mathbf{x}^* \neq \mathbf{x}^l$. Suppose that $P(\mathbf{x}^*) = \emptyset$. Then, $d(\mathbf{x}^*) = \mathbf{x}^l$. Thus,

$$\mathbf{x}^* = h(\mathbf{x}^*) = \lfloor \mathbf{x}^l \rfloor = \mathbf{x}^l.$$

A contradiction occurs. Therefore, $P(\mathbf{x}^*) \neq \emptyset$ and, consequently, $d(\mathbf{x}^*) \in P$. Since

$$\mathbf{x}^* \geq_c d(\mathbf{x}^*) \geq_c \lfloor d(\mathbf{x}^*) \rfloor = h(\mathbf{x}^*) = \mathbf{x}^*,$$

hence, $d(\mathbf{x}^*) = \lfloor d(\mathbf{x}^*) \rfloor = \mathbf{x}^*$. This completes the proof. □

Let $(L, \leq_c)$ be a finite lattice and f an order preserving mapping from L into itself. As a corollary of Theorem 8 and Lemma 3, we obtain that

Corollary 4. *Given lattice $(L, \leq_c)$ and an order preserving mapping f as a polynomial function, determining that f has a unique fixed point in L is a Co-NP hard problem.*

It is a remark that the result can also be applied to derive a fixed-point iterative method for determining wether there is an integer point in such a polytope.

5.2 Co-NP-hard in Lexicographic Ordering

Let P be a polytope of R^n. It is well known that determining there is no integer point in P is a Co-NP hard problem. We assume $n \geq 2$. Let $x^{\max} = (x_1^{\max}, x_2^{\max}, \ldots, x_n^{\max})^\top$ with $x_j^{\max} = \max_{x \in P} x_j$, $j = 1, 2, \ldots, n$, and $x^{\min} = (x_1^{\min}, x_2^{\min}, \ldots, x_n^{\min})^\top$ with $x_j^{\min} = \min_{x \in P} x_j$, $j = 1, 2, \ldots, n$. Then, $x^{\min} \leq_c x \leq_c x^{\max}$ for all $x \in P$. Let $D(P) = \{x \in Z^n \mid x^l \leq_c x \leq_c x^u\}$, where $x^u = \lfloor x^{\max} \rfloor$ and $x^l = \lfloor x^{\min} \rfloor$. Thus, $D(P)$ contains all integer points of P. We assume without loss of generality that $x^l <_c x^{\min}$ (Let $x_i^l = x_i^{\min} - 1$ if $x_i^l = x_i^{\min}$ for some $i \in N$).

For $y \in R^n$ and $k \in N$, let $P(y, k) = \{x \in P \mid x_i = y_i,\ i = 1, 2, \ldots, k\}$.

Definition 6. *For $y \in D(P)$, $h(y) = (h_1(y), h_2(y), \ldots, h_n(y))^\top \in D(P)$ is given in the following procedure:*

Step 0: *If $y_1 = x_1^l$ or $y \in P$, let*

$$h(y) = \begin{cases} x^l & \text{if } y_1 = x_1^l, \\ y & \text{if } y \in P. \end{cases}$$

Otherwise, let $k = 2$ and go to **Step 1**.

Step 1: *Solve linear program*

$$\begin{array}{rl} \min & x_k - v_k \\ \textit{subject to} & x \in P(y, k-1) \text{ and } v \in P(y, k-1), \end{array}$$

to obtain the optimal values (x_k^*, v_k^*). Let $x_k(y) = x_k^*$ and $v_k(y) = v_k^*$. If $\lfloor v_k(y) \rfloor < \lceil x_k(y) \rceil$ or $y_k < \lceil x_k(y) \rceil$, go to **Step 2**. *Otherwise, go to* **Step 3**.

Step 2: *If $y_{k-1} = x_{k-1}^l + 1$,*

$$h_i(y) = \begin{cases} y_i & \text{if } 1 \leq i \leq k-2, \\ x_i^l & \text{if } k-1 \leq i \leq n, \end{cases}$$

$i = 1, 2, \ldots, n$. If $y_{k-1} > x_{k-1}^l + 1$,

$$h_i(y) = \begin{cases} y_i & \text{if } 1 \leq i \leq k-2, \\ y_{k-1} - 1 & \text{if } i = k-1, \\ x_i^u & \text{if } k \leq i \leq n, \end{cases}$$

$i = 1, 2, \ldots, n$.

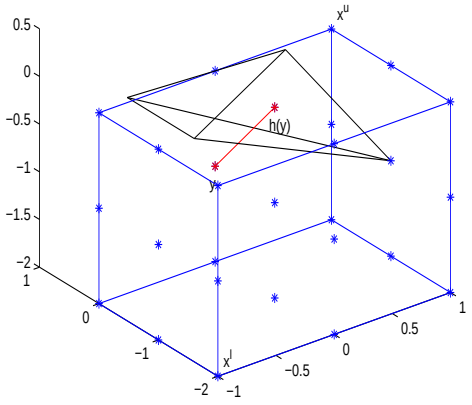


Figure 4: An Illustration of h

Step 3: *If* $y_k > \lfloor v_k(y) \rfloor$,

$$h_i(y) = \begin{cases} y_i & \text{if } 1 \le i \le k-1, \\ \lfloor v_k(y) \rfloor & \text{if } i = k, \\ x_i^u & \text{if } k+1 \le i \le n, \end{cases}$$

$i = 1, 2, \ldots, n$. *Otherwise, let* $k = k + 1$ *and go to* **Step 1**.

Example 2. *Consider* $P = \{x \in R^3 \mid Ax \le b\}$, *where*

$$A = \begin{pmatrix} -1 & 0 & 2 \\ 0 & -2 & 1 \\ -1 & 0 & -2 \\ 1 & 1 & 0 \end{pmatrix}$$

and $b = (0, 1, 1, 0)^\top$. *For* $y = (0, 0, -1)^\top$, $h(y) = (0, -1, 0)$. *An illustration of* h *can be found in Figure 4.*

For any given $y \in D(P)$ with $y_1 \ne x_1^l$ and $y \notin P$, let $k(y)$ denote the value of k at which the procedure determines $h(y)$. Clearly, $k(y) \ge 2$. For simplicity, let $k = k(y)$ from now on.

Lemma 4. $x^l \le h(y) \le_l y$ *and* $h(y) \ne y$ *for all* $y \in D(P)$ *with* $y \ne x^l$ *and* $y \notin P$.

Proof. Clearly, the lemma holds for all $y \in D(P)$ with $y_1 = x_1^l$ and $y \ne x^l$. Let y be any given point in $D(P)$ with $y_1 \ne x_1^l$ and $y \notin P$. It is easy to see that

$$x_i^l < \lceil x_i(y) \rceil \le y_i \le \lfloor v_i(y) \rfloor, \ i = 1, 2, \ldots, k-1.$$

From the procedure, we know that one of the following two situations must occur.

Situation 1: Consider that $\lfloor v_k(y)\rfloor < \lceil x_k(y)\rceil$ or $y_k < \lceil x_k(y)\rceil$.

1. Suppose that $y_{k-1} = x^l_{k-1} + 1$. Then, from **Step 2**, we find that

$$h_i(y) = \begin{cases} y_i & \text{if } 1 \le i \le k-2, \\ x^l_i & \text{if } k-1 \le i \le n, \end{cases}$$

   $i = 1, 2, \ldots, n$. Thus, it follows from $y_{k-1} > x^l_{k-1}$ that $x^l \le h(y) \le y$ and $h(y) \ne y$.

2. Suppose that $y_{k-1} > x^l_{k-1} + 1$. Then, from **Step 2**, we find that

$$h_i(y) = \begin{cases} y_i & \text{if } 1 \le i \le k-2, \\ y_{k-1} - 1 & \text{if } i = k-1, \\ x^u_i & \text{if } k \le i \le n, \end{cases}$$

   $i = 1, 2, \ldots, n$. Thus, it follows from $y_{k-1} - 1 < y_{k-1}$ that $x^l \le h(y) \le_l y$ and $h(y) \ne y$.

Situation 2: Consider that $\lfloor v_k(y)\rfloor \ge \lceil x_k(y)\rceil$ and $y_k > \lfloor v_k(y)\rfloor$. From **Step 3**, we find that

$$h_i(y) = \begin{cases} y_i & \text{if } 1 \le i \le k-1, \\ \lfloor v_k(y)\rfloor & \text{if } i = k, \\ x^u_i & \text{if } k+1 \le i \le n, \end{cases}$$

$i = 1, 2, \ldots, n$. Thus, it follows from $y_k > \lfloor v_k(y)\rfloor$ that $x^l \le h(y) \le_l y$ and $h(y) \ne y$.

These two situations show that $x^l \le h(y) \le_l y$ and $h(y) \ne y$ always hold no matter which occurs. Therefore, the lemma follows immediately. □

As a corollary of Lemma 4, we obtain that

Corollary 5. *For any given $x^* \in D(P)$, $x^* \in P$ if and only if $h(x^*) = x^*$ and $x^* \ne x^l$.*

Theorem 9. *In terms of the lexicographic ordering, h is an increasing mapping from $D(P)$ to itself.*

Proof. Let y^1 and y^2 be any given two different points in $D(P)$ with $y^1 \le_l y^2$. Let q be the index in N satisfying that $y^1_i = y^2_i$, $i = 1, 2, \ldots, q-1$, and $y^1_q < y^2_q$. From the definition of h, we obtain that $h(y^1) = x^l$ if $y^1_1 = x^l_1$ and that $h(y^2) = y^2$ if $y^2 \in P$. Thus, when $y^1_1 = x^l_1$ or $y^2 \in P$, it follows from Lemma 4 that $h(y^1) \le_l h(y^2)$.

Suppose that $y^1_1 \ne x^l_1$ and $y^2 \notin P$. Let $k_1 = k(y^1)$ and $k_2 = k(y^2)$. From the procedure, we know that k_2 is well defined and $k_2 \ge 2$. In the following, we show under four different cases of k_2 that $h(y^1) \le_l h(y^2)$.

Case 1: Consider that $2 \le k_2 \le q-1$. The procedure together with $y^1_i = y^2_i$, $i = 1, 2, \ldots, q-1$, implies that $k_1 = k_2$. Thus, $h(y^1) = h(y^2)$.

Case 2: Consider that $2 \le k_2 = q$. It is easy to see that

$$x^l_i < \lceil x_i(y^2)\rceil \le y^2_i \le \lfloor v_i(y^2)\rfloor,\ i = 1, 2, \ldots, q-1.$$

Since $y^1_i = y^2_i$, $i = 1, 2, \ldots, q-1$, hence,

$$\lceil x_i(y^1)\rceil = \lceil x_i(y^2)\rceil \text{ and } \lfloor v_i(y^1)\rfloor = \lfloor v_i(y^2)\rfloor,\ i = 1, 2, \ldots, q,$$

and

$$x_i^l < \lceil x_i(y^1) \rceil \leq y_i^1 \leq \lfloor v_i(y^1) \rfloor,\ i = 1, 2, \ldots, q-1.$$

From the procedure, we know that one of the following two situations must occur:

Situation 1: Consider that $\lfloor v_q(y^2) \rfloor < \lceil x_q(y^2) \rceil$ or $y_q^2 < \lceil x_q(y^2) \rceil$. From $\lfloor v_q(y^2) \rfloor < \lceil x_q(y^2) \rceil$, $\lceil x_q(y^1) \rceil = \lceil x_q(y^2) \rceil$ and $\lfloor v_q(y^1) \rfloor = \lfloor v_q(y^2) \rfloor$, we derive that $k_1 = k_2 = q$.

1. Suppose that $y_{q-1}^2 = x_{q-1}^l + 1$. Since $y_{q-1}^1 = y_{q-1}^2$, hence, from **Step 2**, we find that

$$h_i(y^2) = \begin{cases} y_i^2 & \text{if } 1 \leq i \leq q-2, \\ x_i^l & \text{if } q-1 \leq i \leq n, \end{cases}$$

   $i = 1, 2, \ldots, n$, and

$$h_i(y^1) = \begin{cases} y_i^1 & \text{if } 1 \leq i \leq q-2, \\ x_i^l & \text{if } q-1 \leq i \leq n, \end{cases}$$

   $i = 1, 2, \ldots, n$. Therefore, $h(y^1) = h(y^2)$.

2. Suppose that $y_{q-1}^2 > x_{q-1}^l + 1$. Since $y_{q-1}^1 = y_{q-1}^2$, hence, from **Step 2**, we find that

$$h_i(y^2) = \begin{cases} y_i^2 & \text{if } 1 \leq i \leq q-2, \\ y_{q-1}^2 - 1 & \text{if } i = q-1, \\ x_i^u & \text{if } q \leq i \leq n, \end{cases}$$

   $i = 1, 2, \ldots, n$, and

$$h_i(y^1) = \begin{cases} y_i^1 & \text{if } 1 \leq i \leq q-2, \\ y_{q-1}^1 - 1 & \text{if } i = q-1, \\ x_i^u & \text{if } q \leq i \leq n, \end{cases}$$

   $i = 1, 2, \ldots, n$. Therefore, $h(y^1) = h(y^2)$.

Situation 2: Consider that $\lfloor v_q(y^2) \rfloor \geq \lceil x_q(y^2) \rceil$ and $y_q^2 > \lfloor v_q(y^2) \rfloor$. From **Step 3**, we find that

$$h_i(y^2) = \begin{cases} y_i^2 & \text{if } 1 \leq i \leq q-1, \\ \lfloor v_q(y^2) \rfloor & \text{if } i = q, \\ x_i^u & \text{if } q+1 \leq i \leq n, \end{cases}$$

$i = 1, 2, \ldots, n$.

- Suppose that $y^1 \in P$. Then, $h(y^1) = y^1$ and

$$\lceil x_i(y^1) \rceil \leq y_i^1 \leq \lfloor v_i(y^1) \rfloor,\ i = 1, 2, \ldots, n.$$

  Thus, from $\lfloor v_q(y^1) \rfloor = \lfloor v_q(y^2) \rfloor$, we obtain that

$$h_q(y^1) = y_q^1 \leq \lfloor v_q(y^2) \rfloor = h_q(y^2).$$

  Therefore, $h(y^1) \leq h(y^2)$.

- Suppose that $y^1 \notin P$. From $y_i^1 = y_i^2$, $i = 1, 2, \ldots, q-1$, and $k_2 = q$, we derive that $k_1 \geq q$.
  1. Assume that $k_1 = q$. Since $\lfloor v_q(y^1) \rfloor \geq \lceil x_q(y^1) \rceil$, hence, either $y_q^1 > \lfloor v_q(y^1) \rfloor$ or $y_q^1 < \lceil x_q(y^1) \rceil$.

(a) Suppose that $y_q^1 > \lfloor v_q(y^1) \rfloor$. Then, from **Step 3**, we obtain that

$$h_i(y^1) = \begin{cases} y_i^1 & \text{if } 1 \le i \le q-1, \\ \lfloor v_q(y^1) \rfloor & \text{if } i = q, \\ x_i^u & \text{if } q+1 \le i \le n, \end{cases}$$

$i = 1, 2, \ldots, n$. Therefore, $h(y^1) = h(y^2)$.

(b) Suppose that $y_q^1 < \lceil x_q(y^1) \rceil$. From **Step 2**, we obtain that, if $y_{q-1}^1 = x_{q-1}^l + 1$, then

$$h_i(y^1) = \begin{cases} y_i^1 & \text{if } 1 \le i \le q-2, \\ x_i^l & \text{otheriwse}, \end{cases}$$

$i = 1, 2, \ldots, n$; and if $y_{q-1}^1 > x_{q-1}^l + 1$, then

$$h_i(y^1) = \begin{cases} y_i^1 & \text{if } 1 \le i \le q-2, \\ y_{q-1}^1 - 1 & \text{if } i = q-1, \\ x_i^u & \text{if } q \le i \le n, \end{cases}$$

$i = 1, 2, \ldots, n$. Therefore, if $y_{q-1}^1 = x_{q-1}^l + 1$, then $h(y^1) \le h(y^2)$; and if $y_{q-1}^1 > x_{q-1}^l + 1$, then $h(y^1) \le_l h(y^2)$ follows from $h_i(y^1) = h_i(y^2)$, $i = 1, 2, \ldots, q-2$, and $h_{q-1}(y^1) = y_{q-1}^1 - 1 < y_{q-1}^1 = y_{q-1}^2 = h_{q-1}(y^2)$.

2. Assume that $k_1 > q$. Then, $k_1 - 1 \ge q$ and

$$\lceil x_i(y^1) \rceil \le y_i^1 \le \lfloor v_i(y^1) \rfloor, \ i = 1, 2, \ldots, k_1 - 1.$$

Thus, from the procedure, we obtain that $h_i(y^1) = y_i^1 = y_i^2 = h_i(y^2)$, $i = 1, 2, \ldots, q-1$, $h_q(y^1) \le y_q^1 \le \lfloor v_q(y^1) \rfloor = \lfloor v_q(y^2) \rfloor = h_q(y^2)$, and $h_i(y^1) \le x_i^u = h_i(y^2)$, $i = q+1, q+2, \ldots, n$. Therefore, $h(y^1) \le h(y^2)$.

Case 3: Consider that $2 \le k_2 = q + 1$. It is easy to see that

$$x_i^l < \lceil x_i(y^2) \rceil \le y_i^2 \le \lfloor v_i(y^2) \rfloor, \ i = 1, 2, \ldots, q.$$

Since $y_i^1 = y_i^2$, $i = 1, 2, \ldots, q-1$, hence,

$$\lceil x_i(y^1) \rceil = \lceil x_i(y^2) \rceil \text{ and } \lfloor v_i(y^1) \rfloor = \lfloor v_i(y^2) \rfloor, \ i = 1, 2, \ldots, q,$$

$$x_i^l < \lceil x_i(y^1) \rceil \le y_i^1 \le \lfloor v_i(y^1) \rfloor, \ i = 1, 2, \ldots, q-1,$$

and

$$\lceil x_q(y^1) \rceil \le \lfloor v_q(y^1) \rfloor.$$

From the procedure, we know that one of the following two situations must occur:

Situation 1: Consider that $\lfloor v_{q+1}(y^2) \rfloor < \lceil x_{q+1}(y^2) \rceil$ **or** $y_{q+1}^2 < \lceil x_{q+1}(y^2) \rceil$.

1. Suppose that $y_q^2 = x_q^l + 1$. From **Step 2**, we find that

$$h_i(y^2) = \begin{cases} y_i^2 & \text{if } 1 \le i \le q-1, \\ x_i^l & \text{if } q \le i \le n, \end{cases}$$

$i = 1, 2, \ldots, n$. From $y_q^1 < y_q^2 = x_q^l + 1$, we get that $y_q^1 = x_q^l$ and $k_1 = q \geq 2$ because $y_1^1 \neq x_1^l$. Thus, it follows from $y_q^1 = x_q^l < \lceil x_q(y^1) \rceil$ and **Step 2** that, if $y_{q-1}^1 = x_{q-1}^l + 1$, then

$$h_i(y^1) = \begin{cases} y_i^1 & \text{if } 1 \leq i \leq q-2, \\ x_i^l & \text{if } q-1 \leq i \leq n, \end{cases}$$

$i = 1, 2, \ldots, n$; and if $y_{q-1}^1 > x_{q-1}^l + 1$, then

$$h_i(y^1) = \begin{cases} y_i^1 & \text{if } 1 \leq i \leq q-2, \\ y_{q-1}^1 - 1 & \text{if } i = q-1, \\ x_i^u & \text{if } q \leq i \leq n, \end{cases}$$

$i = 1, 2, \ldots, n$. Therefore, if $y_{q-1}^1 = x_{q-1}^l + 1$, then $h(y^1) \leq h(y^2)$; and if $y_{q-1}^1 > x_{q-1}^l + 1$, then $h(y^1) \leq_l h(y^2)$ follows from $h_i(y^1) = h_i(y^2)$, $i = 1, 2, \ldots, q-2$, and $h_{q-1}(y^1) = y_{q-1}^1 - 1 < y_{q-1}^2 = h_{q-1}(y^2)$.

2. Suppose that $y_q^2 > x_q^l + 1$. From **Step 2**, we find that

$$h_i(y^2) = \begin{cases} y_i^2 & \text{if } 1 \leq i \leq q-1, \\ y_q^2 - 1 & \text{if } i = q, \\ x_i^u & \text{if } q+1 \leq i \leq n, \end{cases}$$

$i = 1, 2, \ldots, n$.

- Suppose that $y^1 \in P$. Then, $h(y^1) = y^1$. Thus, $h(y^1) \leq h(y^2)$.
- Suppose that $y^1 \notin P$. Then, $k_1 \geq q$.
  (a) Assume that $k_1 = q$. Since $\lceil x_q(y^1) \rceil \leq \lfloor v_q(y^1) \rfloor$, hence, either $y_q^1 > \lfloor v_q(y^1) \rfloor$ or $y_q^1 < \lceil x_q(y^1) \rceil$.
    i. Suppose that $y_q^1 > \lfloor v_q(y^1) \rfloor$. Then, from **Step 3**, we obtain that

$$h_i(y^1) = \begin{cases} y_i^1 & \text{if } 1 \leq i \leq q-1, \\ \lfloor v_q(y^1) \rfloor & \text{if } i = q, \\ x_i^u & \text{if } q+1 \leq i \leq n, \end{cases}$$

    $i = 1, 2, \ldots, n$. Thus, $h_q(y^1) < y_q^1$. Therefore, $h(y^1) \leq h(y^2)$ because $y_q^1 < y_q^2$.
    ii. Suppose that $y_q^1 < \lceil x_q(y^1) \rceil$. From **Step 2**, we obtain that, if $y_{q-1}^1 = x_{q-1}^l + 1$, then

$$h_i(y^1) = \begin{cases} y_i^1 & \text{if } 1 \leq i \leq q-2, \\ x_i^l & \text{otheriwse}, \end{cases}$$

    $i = 1, 2, \ldots, n$; and if $y_{q-1}^1 > x_{q-1}^l + 1$, then

$$h_i(y^1) = \begin{cases} y_i^1 & \text{if } 1 \leq i \leq q-2, \\ y_{q-1}^1 - 1 & \text{if } i = q-1, \\ x_i^u & \text{if } q \leq i \leq n, \end{cases}$$

    $i = 1, 2, \ldots, n$. Therefore, if $y_{q-1}^1 = x_{q-1}^l + 1$, then $h(y^1) \leq h(y^2)$; and if $y_{q-1}^1 > x_{q-1}^l + 1$, then $h(y^1) \leq_l h(y^2)$ follows from $h_i(y^1) = h_i(y^2)$, $i = 1, 2, \ldots, q-2$, and $h_{q-1}(y^1) = y_{q-1}^1 - 1 < y_{q-1}^2 = h_{q-1}(y^2)$.

(b) Assume that $k_1 > q$. From the procedure, we derive that $h_i(y^1) = y_i^1$, $i = 1, 2, \ldots, q-1$, and $h_q(y^1) \leq y_q^1$. Thus, $h(y^1) \leq h(y^2)$ because $y_q^1 < y_q^2$.

Situation 2: Consider that $\lfloor v_{q+1}(y^2) \rfloor \geq \lceil x_{q+1}(y^2) \rceil$ and $y_{q+1}^2 > \lfloor v_{q+1}(y^2) \rfloor$. From **Step 3**, we find that

$$h_i(y^2) = \begin{cases} y_i^2 & \text{if } 1 \leq i \leq q, \\ \lfloor v_{q+1}(y^2) \rfloor & \text{if } i = q+1, \\ x_i^u & \text{if } q+2 \leq i \leq n, \end{cases}$$

$i = 1, 2, \ldots, n$.

- Suppose that $y^1 \in P$. Then, $h(y^1) = y^1$. Thus, from $y_q^1 < y_q^2$, we obtain that $h_q(y^1) < y_q^2 = h_q(y^2)$. Therefore, $h(y^1) \leq_l h(y^2)$ follows immediately from $h_i(y^1) = h_i(y^2)$, $i = 1, 2, \ldots, q-1$, and $h_q(y^1) < h_q(y^2)$.
- Suppose that $y^1 \notin P$. Then, $k_1 \geq q$.
  1. Assume that $k_1 = q$. Since $\lceil x_q(y^1) \rceil \leq \lfloor v_q(y^1) \rfloor$, hence, either $y_q^1 > \lfloor v_q(y^1) \rfloor$ or $y_q^1 < \lceil x_q(y^1) \rceil$.
     (a) Suppose that $y_q^1 > \lfloor v_q(y^1) \rfloor$. From **Step 3**, we obtain that

$$h_i(y^1) = \begin{cases} y_i^1 & \text{if } 1 \leq i \leq q-1, \\ \lfloor v_q(y^1) \rfloor & \text{if } i = q, \\ x_i^u & \text{if } q+1 \leq i \leq n, \end{cases}$$

     $i = 1, 2, \ldots, n$. Thus, $h_q(y^1) < y_q^1$. Therefore, $h(y^1) \leq_l h(y^2)$ follows from $h_i(y^1) = h_i(y^2)$, $i = 1, 2, \ldots, q-1$, and $h_q(y^1) < y_q^1 < y_q^2 = h_q(y^2)$.
     (b) Suppose that $y_q^1 < \lceil x_q(y^1) \rceil$. From **Step 2**, we obtain that, if $y_{q-1}^1 = x_{q-1}^l + 1$, then

$$h_i(y^1) = \begin{cases} y_i^1 & \text{if } 1 \leq i \leq q-2, \\ x_i^l & \text{otheriwse}, \end{cases}$$

     $i = 1, 2, \ldots, n$; and if $y_{q-1}^1 > x_{q-1}^l + 1$, then

$$h_i(y^1) = \begin{cases} y_i^1 & \text{if } 1 \leq i \leq q-2, \\ y_{q-1}^1 - 1 & \text{if } i = q-1, \\ x_i^u & \text{if } q \leq i \leq n, \end{cases}$$

     $i = 1, 2, \ldots, n$. Therefore, if $y_{q-1}^1 = x_{q-1}^l + 1$, then $h(y^1) \leq h(y^2)$; and if $y_{q-1}^1 > x_{q-1}^l + 1$, then $h(y^1) \leq_l h(y^2)$ follows from $h_i(y^1) = h_i(y^2)$, $i = 1, 2, \ldots, q-2$, and $h_{q-1}(y^1) = y_{q-1}^1 - 1 < y_{q-1}^1 = y_{q-1}^2 = h_{q-1}(y^2)$.
  2. Assume that $k_1 > q$. From the procedure, we derive that $h_i(y^1) = y_i^1$, $i = 1, 2, \ldots, q-1$, and $h_q(y^1) \leq y_q^1$. Thus, $h(y^1) \leq_l h(y^2)$ follows immediately from $h_i(y^1) = h_i(y^2)$, $i = 1, 2, \ldots, q-1$, and $h_q(y^1) \leq y_q^1 < y_q^2 = h_q(y^2)$.

Case 4: Consider that $k_2 > q+1$. From $k_2 - 1 > q$, we obtain that $h_i(y^2) = y_i^2$, $i = 1, 2, \ldots, q$. Thus, $y^1 \leq_l h(y^2)$ since $y_i^1 = y_i^2$, $i = 1, 2, \ldots, q-1$, and $y_q^1 < y_q^2$. Therefore, it follows immediately from Lemma 4 that $h(y^1) \leq_l h(y^2)$.

The above four cases show that $h(y^1) \leq_l h(y^2)$ always holds no matter which occurs. This completes the proof. □

From Definition 6, one can see that, for each $y \in D(P)$, it takes at most n linear programs to determine $h(y)$. Therefore, $h(y)$ is polynomial-time defined for any given $y \in D(P)$.

As a corollary of Theorem 9, we obtain that

Corollary 6. *Given lattice $(L, \leq_l)$ and an order preserving mapping f as a polynomial function, determining that f has a unique fixed point in L is a Co-NP hard problem.*

We remark that the result can also be used to obtain a fixed-point iterative method for determining whether there is an integer point in a polytope.

Next, we give a simple but naive reduction.

Definition 7. *For $z \in D(P)$, $h(z) = (h_1(z), h_2(z), \ldots, h_n(z))^\top \in D(P)$ is given as follows:*

Step 0: *If $z \in P$ or $z = x^l$, let $h(z) = z$. Otherwise, let $k = n$ and go to* **Step 1**.

Step 1: *If $z_k > x_k^l$, let $h(z) = (h_1(z), h_2(z), \ldots, h_n(z))^\top$ with*

$$h_i(z) = \begin{cases} z_i & \text{if } i < k, \\ z_k - 1 & \text{if } i = k, \\ x_i^u & \text{if } i > k, \end{cases}$$

$i = 1, 2, \ldots, n$. Otherwise, go to **Step 2**.

Step 2: *Let $k = k - 1$ and go to* **Step 1**.

Clearly, $h(z)$ is polynomial-time defined for any $z \in D(P)$. Furthermore, it is easy to see that

Theorem 10. *In terms of the lexicographic ordering, h is an increasing mapping from $D(P)$ to itself. Moreover, $h(z^*) = z^* \neq x^l$ if and only if $z^* \in P$.*

6 Conclusion and Open Problems

Results on the Tarski's fixed points contrast with the past results for the general fixed point computation in several ways. First, in the oracle function model, several fixed point computational problems are known to require an exponential number of queries for constant dimensions, including the two dimensional case (Chen et al., Deng et al. and Hirsch et al.)[10, 13, 17]. Our results prove the Tarski's fixed point to be polynomial in the oracle function model with constant dimensions. They also show that it is so for the polynomial function model, which is also different from those fixed point computational problems which are known to be PPAD-complete for constant dimensions, including the two dimensional case (Chen et al. and Deng et al.)[9, 13]. In the polynomial function model, we have proved that determining the uniqueness is co-NP-complete. In comparison, the uniqueness of Nash equilibrium is known to be co-NP-complete but its existence is in PPAD.

The above comparisons with the previous work leave the following outstanding open problem: Is it PPAD-complete to find a Tarski's fixed point in the variable dimension n for the polynomial function model ? This problem is known to be true for finding a Sperner simplex in dimension n with n as a variable. We conjecture that this is also true for finding a Tarski's fixed point.

References

[1] A. Blum, J. Jackson, T. Sandholm, M. Zinkevich (2004). Preference elicitation and query learning. Journal of Machine Learning Research (JMLR), 5:649–667.

[2] F. Bernstein and A. Federgruen (2005). Decentralized supply chains with competing retailers under demand uncertainty, Management Science 51: 18-29.

[3] F. Bernstein and A. Federgruen (2004). A general equilibrium model for industries with price and service competition, Operations Research 52: 868-886.

[4] L. Blumrosen, N. Nisan. Combinatorial Auctions (2007). In Algorithmic Game Theory (N. Nisan, T. Roughgarden, E. Tardos, V. Vazirani, eds.), Cambridge University Press, London.

[5] B. Codenotti, A. Saberi, K. R. Varadarajan and Y. Ye (2008). The complexity of equilibria: Hardness results for economies via a correspondence with games. Theor. Comput. Sci. 408(2-3): 188-198.

[6] G.P. Cachon (2001). Stcok wars: inventory competition in a two echelon supply chain, Operations Research 49: 658-674.

[7] G.P. Cachon and M.A. Lariviere (1999). Capacity choice and allocation: strategic behavior and supply chain performance, Management Sceince 45: 1091-1108.

[8] C.L. Chang, Y.D. Lyuu and Y.W. Ti (2008). The complexity of Tarski's fixed point theorem, Theoretical Computer Science 401: 228-235.

[9] X. Chen and X. Deng (2009). On the Complexity of 2D Discrete Fixed Point Problem, Theoretical Computer Science **410**(44), 4448–4456.

[10] X. Chen and X. Deng (2008). Matching algorithmic bounds for finding a Brouwer fixed point, Journal of the ACM **55**(3), 13:1–13:26.

[11] X. Chen, X. Deng and S. Teng (2008). Settling the computational complexity of two player Nash equilibrium, Journal of the ACM (JACM) 56(3).

[12] S. Dobzinski, N. Nisan, M. Schapira (2010). Approximation Algorithms for Combinatorial Auctions with Complement-Free Bidders. Mathematics of Operations Research, February; 35: 1–13.

[13] X. Deng, Q. Qi, A. Saberi, J. Zhang (2011). Discrete Fixed Points: Models, Complexities and Applications, accepted by MOR

[14] F. Echenique (2007). Finding all equilibria in games of strategic complements, Journal of Economic Theory 135: 514-532.

[15] D. Fudenberg and J. Tirole (1991). Game Theory, MIT Press.

[16] I. Gilboa and E. Zemmel (1989). Nash and Correlated Equilibria: Some Complexity Considerations, Games and Economic Behavior 1: 80-93.

[17] M.D. Hirsch, C.H. Papadimitriou and S. Vavasis (1989). Exponential Lower Bounds for Finding Brouwer Fixed Points, Journal of Complexity, Vol. 5, pp.379–416.

[18] J.C. Lagarias (1985). The computational complexity of simultaneous Diophantine approximation problems, SIAM Journal on Computing 14: 196-209.

[19] S.A. Lippman and K.F. McCardle (1997). The competitive newsboy, Operations Research 45: 54-65.

[20] P. Milgrom and J. Roberts (1990). Rationalizability, learning, and equilibrium in games with strategic complementarities, Econometrica 58: 155-1277.

[21] P. Milgrom and J. Roberts (1994). Comparing equilibria, American Economic Review 84: 441-459.

[22] P. Milgrom and C. Shannon (1994). Monotone comparative statics, Econometrica 62: 157-180.

[23] A. Tarski (1955). A lattice-theoretical fixpoint theorem and its applications, Pacific Journal of Mathematics 5: 285-308.

[24] D.M. Topkis (1979). Equilibrium points in nonzero-sum n-person submodular games, SIAM Journal on Control and Optimization 17: 773-787.

[25] D.M. Topkis (1998). Supermodularity and Complementarity, Princeton University Press.

[26] X. Vives (1990). Nash equilibrium with strategic complementarities, Journal of Mathematical Economics 19: 305-321.

[27] X. Vives (2005). Complemetarities and games: new developments, Journal of Economic Literature XLIII: 437-479.