Tarski's Theorem, Supermodular Games, and the Complexity of Equilibria

Kousha Etessami
School of Informatics, University of Edinburgh, UK
kousha@inf.ed.ac.uk

Christos Papadimitriou
Dept. of Computer Science, Columbia University, NY, USA
christos@cs.columbia.edu

Aviad Rubinstein
Dept. of Computer Science, Stanford University, CA, USA
aviad@cs.stanford.edu

Mihalis Yannakakis
Dept of Computer Science, Columbia University, NY, USA
mihalis@cs.columbia.edu

Abstract

The use of monotonicity and Tarski's theorem in existence proofs of equilibria is very widespread in economics, while Tarski's theorem is also often used for similar purposes in the context of verification. However, there has been relatively little in the way of analysis of the complexity of finding the fixed points and equilibria guaranteed by this result. We study a computational formalism based on monotone functions on the d-dimensional grid with sides of length N, and their fixed points, as well as the closely connected subject of supermodular games and their equilibria. It is known that finding some (any) fixed point of a monotone function can be done in time $\log^d N$, and we show it requires at least $\log^2 N$ function evaluations already on the 2-dimensional grid, even for randomized algorithms. We show that the general Tarski problem of finding some fixed point, when the monotone function is given succinctly (by a boolean circuit), is in the class PLS of problems solvable by local search and, rather surprisingly, also in the class PPAD. Finding the greatest or least fixed point guaranteed by Tarski's theorem, however, requires $d \cdot N$ steps, and is NP-hard in the white box model. For supermodular games, we show that finding an equilibrium in such games is essentially computationally equivalent to the Tarski problem, and finding the maximum or minimum equilibrium is similarly harder. Interestingly, two-player supermodular games where the strategy space of one player is one-dimensional can be solved in $O(\log N)$ steps. We also show that computing (approximating) the value of Condon's (Shapley's) stochastic games reduces to the Tarski problem. An important open problem highlighted by this work is proving a $\Omega(\log^d N)$ lower bound for small fixed dimension $d \geq 3$.

2012 ACM Subject Classification Theory of computation → Computational complexity and cryptography

Keywords and phrases Tarski's theorem, supermodular games, monotone functions, lattices, fixed points, Nash equilibria, computational complexity, PLS, PPAD, stochastic games, oracle model, lower bounds

Digital Object Identifier 10.4230/LIPIcs.ITCS.2020.18

Related Version A full version [11] is available at: `https://arxiv.org/abs/1909.03210`.

Funding This work was partially supported by NSF Grants CCF-1703295, CCF-1763970.

Acknowledgements Thanks to Alexandros Hollender for pointing us to [2].

1 Introduction

Equilibria are paramount in economics, because guaranteeing their existence in a particular strategic or market-like framework enables one to consider "What happens at equilibrium?" without further analysis. Equilibrium existence theorems are nontrivial to prove. The best known example is Nash's theorem [18], whose proof in 1950, based on Brouwer's fixed point theorem, transformed game theory, and inspired the Arrow-Debreu price equilibrium results [1], among many others. Decades later, complexity analysis of these theorems and corresponding solution concepts by computer scientists has created a fertile and powerful field of research [19].

Not all equilibrium theorems in economics, however, rely on Brouwer's fixed point theorem for their proof (even though, in a specific sense made clear and proved in this paper, they could have...). Many of the exceptions ultimately rely on *Tarski's fixed point theorem* [22], stating that all monotone functions on a complete lattice have a fixed point – and in fact a whole sublattice of fixed points with a largest and smallest element [23, 17, 24]. In contrast to the equilibrium theorems whose proof relies on Brouwer's fixed point theorem, there has been relatively little complexity analysis of Tarski's fixed point theorem and the equilibrium results it enables. (We discuss prior related work at the end of this introduction.)

Here we present several results in this direction. Let $[N] = \{1, \ldots, N\}$. To formulate the basic problem, we consider a monotone function f on the d-dimensional grid $[N]^d$, that is, a function $f : [N]^d \mapsto [N]^d$ such that for all $x, y \in [N]^d$, $x \geq y$ implies $f(x) \geq f(y)$; in the black-box oracle model, we can query this function with specific vectors $x \in [N]^d$; in the white-box model we assume that the function is presented by a boolean circuit[1]. Thus, d and N are the basic parameters to our model; it is useful to think of d as the dimensionality of the problem, while N is something akin to the inverse of the desired approximation ϵ.

- Tarski's theorem in the grid framework is easy to prove. Let $\bar{1} = (1, \ldots, 1)$ denote the (d-dimensional) all-1 vector. Consider the sequence of grid points $\bar{1}, f(\bar{1}), f(f(\bar{1})), \ldots, f^i(\bar{1}), \ldots$. From monotonicity of f, by induction on i we get, for all $i \geq 0$, $f^i(\bar{1}) \leq f^{i+1}(\bar{1})$. Unless a fixed point is arrived at, the sum of the coordinates must increase at each iteration. Therefore, after at most dN iterations of f applied to $\bar{1}$, a fixed point is found. In other words $f^{dN}(\bar{1}) = f^{dN+1}(\bar{1})$.
- This immediately suggests an $O(dN)$ algorithm. But an $O(\log^d N)$ algorithm is also known[2]: Consider the $(d-1)$-dimensional function obtained by fixing the "input value" in the d'th coordinate of the function f with some value r_d (initialize $r_d := \lceil N/2 \rceil$). Find a fixed point $z^* \in [N]^{d-1}$ of this $(d-1)$-dimensional monotone function $f(z, r_d)$ (recursively). If the dth coordinate $f_d(z^*, r_d)$ of $f(z^*, r_d)$, is equal to r_d, then (z^*, r_d) is a fixed point of the overall function f, and we are done. Otherwise, a binary search on the d'th coordinate is enabled: we need to look for a larger (smaller) value of r_d if $f_d(z^*, r_d) > r_d$ (respectively, if $f_d(z^*, r_d) < r_d$). By an easy induction, this establishes the $O(\log^d N)$ upper bound ([5]).
- We conjecture that this algorithm is essentially optimal in the black box sense, for small fixed constant dimension d. In Theorem 7 we prove this result for the $d = 2$ case. We provide a class of monotone functions that we call *the herringbones*: two monotonic paths,

[1] Naturally, one could have addressed the more general problem in which the lattice is itself presented in a general way through two functions *meet* and *join;* however, this framework (a) leads quickly and easily to intractability; and (b) does not capture any more applications in economics than the one treated here.

[2] This algorithm appears to have been first observed in [5].

one starting from $\bar{1}$ and the other from $\bar{N}$, meeting at the fixed point, while all other points in the $N \times N$ grid are mapped diagonally: $f(x) = x + (-1, +1)$ or $x + (+1, -1)$, whichever of the points is closer to the monotonic path that contains the fixed point. We prove that any randomized algorithm needs to make $\Omega(\log^2 N)$ queries (in expectation) to find the fixed point.

- Can this lower bound result be generalized to fixed $d \geq 3$? This is a key question left open by this paper. There are several obstacles to a proof establishing, e.g., a $\Omega(\log^3 N)$ lower bound in the 3-dimensional case ($d = 3$), and some possible ways for overcoming them. First, it is not easy to identify a suitable "herringbone-like" function in three or more dimensions – a monotone family of functions built around a path from $\bar{1}$ to $\bar{N}$. It nevertheless seems plausible that $\log^d N$ should still be (close to) a lower bound on any such algorithm (assuming of course that N is sufficiently larger than d, so that the dN algorithm does not violate the lower bound). We prove one encouraging result in this context: We give an alternative proof of the $d = 2$ lower bound, in which we establish that any *deterministic* black-box algorithm for TARSKI in two dimensions *must solve a sequence of* $\Omega(\log N)$ *one-dimensional problems* (Theorem 13), a result pointing to a possible induction on d (recall that this is precisely the form of the $\log^d N$ algorithm).
- Tarski's theorem further asserts that there is a greatest and a least fixed point, and these fixed points are especially useful in the economic applications of the result (see for example [17]). It is not hard to see, however, that finding these fixed points is **NP**-hard, and takes $\Omega(dN)$ time in the black box model (see Proposition 1).
- In terms of complexity classes, the problem TARSKI is obviously in the class **TFNP** of total function (total search) problems. But where exactly? We show (Theorem 4) that it belongs in the class PLS of local optimum search problems.
- Surprisingly, TARSKI is also in the class P^{PPAD} of problems reducible to a Brouwer fixed point problem (Theorem 5), and thus, by the known fact that the class PPAD is closed under polynomial time Turing reductions ([2]) it is in PPAD (Corollary 6). This result presents a heretofore unsuspected connection between two main sources of equilibrium results in economics.
- *Supermodular games* [23, 17, 24] – or games with strategic complementarities – comprise a large and important class of economic models, with complete lattices as strategy spaces, in which a player's best response is a monotone function (or monotone correspondence) of the other player's strategies. They always have pure Nash equilibria due to Tarski's theorem. We show that finding an equilibrium for a supermodular game with (discrete) Euclidean grid strategy spaces is essentially computationally equivalent to the problem of finding a Tarski fixed point of a monotone map (Proposition 14 and Theorem 16). If there are two players and one of them has a one-dimensional strategy space, we show that a Nash equilibrium can be found in logarithmic time (in the size of the strategy spaces).
- *Stochastic games* [21, 4]. We show that the problems of computing the (irrational) value of Shapley's discounted stochastic games to desired accuracy, and computing the exact value of Condon's simple stochastic games (SSG), are both P-time reducible to the Tarski problem. The proofs employ known characterizations of the value of both Shapley's stochastic games and Condon's SSGs in terms of monotone fixed point equations, which can also be viewed as monotone "polynomially contracting" maps with a unique fixed point, and from properties of polynomially contracting maps, see [12].

Prior related work. In recent years a number of technical reports and papers by Dang, Qi, and Ye, have considered the complexity of computational problems related to Tarski's theorem [5, 7, 6]. In particular, in [5] the authors provided the already-mentioned $\log^d N$

algorithm for computing a Tarski fixed point for a discrete map, $f : [N]^d \to [N]^d$, which is monotone under the coordinate-wise order. In [5] they also establish that determining the uniqueness of the fixed point of a monotone map under coordinate-wise order is coNP-hard, and that uniqueness under lexicographic order is also coNP-hard (already in one dimension). In [7] the authors studied another variant of the Tarski problem, namely computing another fixed point of a monotone function in an expanded domain where the smallest point is a fixed point; this variant is NP-hard (the claim in the paper that this problem is in PPA has been withdrawn by the authors [8]). In earlier work, Echenique [10], studied algorithms for computing all pure Nash equilibria in supermodular games (and games with strategic complementaries) whose strategy spaces are discrete grids. Of course computing all pure equilibria is harder than computing *some* pure equilibrium; indeed, we show that computing the least (or greatest) pure equilibrium of such a supermodular game is already NP-hard (Corollary 17). In earlier work Chang, Lyuu, and Ti [3] considered the complexity of Tarski's fixed point theorem over a general finite lattice given via an oracle for its partial order (not given it explicitly) and given an oracle for the monotone function, and they observed that the total number of oracle queries required to find some fixed point in this model is linear in the number of elements of the lattice. They did not study monotone functions on euclidean grid lattices, and their results have no bearing on this setting.

Organization of the paper. The rest of the paper is organized as follows. Section 2 provides basic definitions on lattices and monotone functions, and presents some simple basic results. In Section 3 we define the TARSKI problem and show that it is in PLS and PPAD. Section 4 proves the lower bound of $\log^2 N$ on black-box algorithms. Section 5 concerns supermodular games. Section 6 reduces Condon's and Shapley's stochastic games to the TARSKI problem. Finally, Section 7 concludes and discusses open problems. Several of the proofs are only sketched or omitted; more details can be found in the full version of the paper [11].

2 Basics

A partial order $(L, \leq)$ is a *complete lattice* if every nonempty subset S of L has a least upper bound (or supremum or join, denoted $\sup S$ or $\vee S$) and a greatest lower bound (or infimum or meet, denoted $\inf S$ or $\wedge S$) in L. A function $f : L \to L$ is *monotone* if for all pairs of elements $x, y \in L$, $x \leq y$ implies $f(x) \leq f(y)$. A point $x \in L$ is a *fixed point* of f if $f(x) = x$. Tarski's theorem ([22]) states that the set $Fix(f)$ of fixed points of f is a nonempty complete lattice under the same partial order $\leq$; in particular, f has a greatest fixed point (GFP) and a least fixed point (LFP).

In this paper we will take as our underlying lattice L a finite discrete Euclidean grid, which we fix for simplicity to be the integer grid $[N]^d$, for some positive integers N, d, where $[N] = \{1, \ldots, N\}$. Comparison of points is componentwise, i.e. $x \leq y$ if $x_i \leq y_i$ for all $i = 1, \ldots, d$. We will also consider the corresponding continuous box, $[1, N]^d$ that includes all real points in the box. Both, the discrete and continuous box are clearly complete lattices.

Given a monotone function f on the integer grid $[N]^d$, the problem is to compute a fixed point of f (any point in $Fix(f)$). A generally harder problem is to compute specifically the LFP of f or the GFP of f. We consider mostly the oracle model, in which the function f is given by a black-box oracle, and the complexity of the algorithm is measured in terms of the number of queries to the oracle. Alternatively, we can consider also an explicit model in which f is given explicitly by a polynomial-time algorithm (a polynomial-size Boolean circuit), and then the complexity of the algorithm is measured in the ordinary Turing model. Note

that the number of bits needed to represent a point in the domain is $d \log N$, so polynomial time here means polynomial in d and $n = \log N$. The number N^d of points in the domain is exponential.

Tarski's value iteration algorithm provides a simple way to compute the LFP of f: Starting from the lowest point of the lattice, which here is the all-1 vector 1, apply repeatedly f. This generates a monotonically increasing sequence of points $1 \leq f(1) \leq f^2(1) \leq \ldots$ until a fixed point is reached, which is the LFP of f. In every step of the sequence, at least one coordinate is strictly increased, therefore a fixed point is reached in at most $(N-1)d$ steps. In the worst case, the process may take that long, which is exponential in the bit size $d \log N$. Similarly, the GFP can be computed by applying repeatedly f starting from the highest point of the lattice, i.e., from the all-N point, until a fixed point is reached.

Another way to compute some fixed point of a monotone function f (not necessarily the LFP or the GFP) is by a divide-and-conquer algorithm. In one dimension, we can use binary search: If the domain is the set $L(l,h) = \{x \in Z | l \leq x \leq h\}$ of integers between the lowest point l and the highest point h, then compute the value of f on the midpoint $m = (l+h)/2$. If $f(m) = m$ then m is a fixed point; if $f(m) < m$ then recurse on the lower half $L(l,m)$, and if $f(m) > m$ then recurse on the upper half $L(m,h)$. The monotonicity of f implies that f maps the respective half interval into itself. Hence the algorithm correctly finds a fixed point in at most $\log N$ iterations, where N is the number of points.

In the general d-dimensional case, suppose that the domain is the set of integer points in the box defined by the lowest point l and the highest point h, i.e. $L(l,h) = \{x \in Z^d | l \leq x \leq h\}$. Consider the set of points with d-th coordinate equal to $m = (l+h)/2$; their first $d-1$ coordinates induce a $(d-1)$-dimensional lattice $L'(l,h) = \{x \in Z^{d-1} | l_i \leq x_i \leq h_i, i = 1, \ldots d-1\}$. Define the function g on $L'(l,h)$ by letting $g(x)$ consist of the first $d-1$ components of $f(x,m)$. It is easy to see that g is a monotone function on $L'(l,h)$. Recursively, compute a fixed point x^* of g. If $f_d(x^*,m) = m$, then (x^*,m) is a fixed point of f (this holds in particular if $l = h$). If $f_d(x^*,m) > m$, then recurse on $L(f(x^*,m),h)$. If $f_d(x^*,m) < m$, then recurse on $L(l,f(x^*,m))$. In either case, monotonicity implies that if the algorithm recurses, then f maps the smaller box into itself and thus has a fixed point in it. An easy induction shows that the complexity of this algorithm is $O((\log N)^d)$, ([5]).

Computing the least or the greatest fixed point is in general hard, even in one dimension, both in the oracle and in the explicit model.

▶ **Proposition 1.** *Computing the LFP or the GFP of an explicitly given polynomial-time monotone function in one dimension is NP-hard. In the oracle model, the problem requires $\Omega(N)$ queries for a domain of size N.*

Proof. We prove the claim for the LFP; the GFP is similar. Reduction from Satisfiability. Given a Boolean formula ϕ in n variables, let the domain $D = \{0, 1, \ldots, 2^n\}$, and define the function f as follows. For $x \leq 2^n - 1$, viewing x as an n-bit binary number, it corresponds to an assignment to the n variables of ϕ; let $f(x) = x$ if the assignment x satisfies ϕ, and let $f(x) = x+1$ otherwise. Define $f(2^n) = 2^n$. Clearly f is a monotone function and it can be computed in polynomial time. If ϕ is not satisfiable then the LFP of f is 2^n, while if ϕ is satisfiable then the LFP is not 2^n.

For the oracle model, use the same domain D and let f map every $x \leq 2^n - 1$ to x or $x+1$, and $f(2^n) = 2^n$. The LFP is not 2^n iff there exists an $x \leq 2^n - 1$ such that $f(x) = x$, which in the oracle model requires trying all possible $x \leq 2^n - 1$. ◀

In the case of a continuous domain $[1,N]^d$, we may not be able to compute an exact fixed point, and thus we have to be content with approximation. Given an $\epsilon > 0$, an ϵ-approximate fixed point is a point x such that $|f(x) - x| \leq \epsilon$, where we use the L_∞ (max)

norm, i.e. $|f(x) - x| = \max\{|f_i(x) - x_i| | i = 1, \ldots, d\}$. In this context, polynomial time means polynomial in $\log N, d$, and $\log(1/\epsilon)$ (the number of bits of the approximation). An ϵ-approximate fixed point need not be close to any actual fixed point of f. A problem that is generally harder is to compute a point that approximates some actual fixed point, and an even harder task is to approximate specifically the LFP or the GFP of f. Tarski's value iteration algorithm, starting from the lowest point converges in the limit to the LFP (and if started from the highest point, it converges to the GFP), but there is no general bound on the number of iterations needed to get within ϵ of the LFP (or the GFP). The algorithm reaches however an ϵ-approximate fixed point within Nd/ϵ iterations (note, this is exponential in $\log N, \log(1/\epsilon)$).

It is easy to see that the approximate fixed point problem for the continuous case reduces to the exact fixed point problem for the discrete case.

▶ **Proposition 2.** *The problem of computing an ϵ-approximate fixed point of a given monotone function on the continuous domain $[1, N]^d$ reduces to the exact fixed point problem on a discrete domain $[N/\epsilon]^d$.*

Proof. Given the monotone function f on the continuous domain $D_1 = [1, N]^d$, consider the discrete domain $D_2 = \{x \in Z^d | k \leq x_i \leq Nk, i = 1, \ldots, d\}$, where $k = \lceil 1/\epsilon \rceil$, and define the function g on D_2 as follows. For every $x \in D_2$, let $g(x)$ be obtained from $kf(x/k)$ by rounding each coordinate to the nearest integer, with ties broken (arbitrarily) in favor of the ceiling. Since f is monotone, g is also monotone. If x^* is a fixed point of g, then $kf(x^*/k)$ is within $1/2$ of x^* in every coordinate, and hence $f(x^*/k)$ is within $1/2k < \epsilon$ of x^*/k. Thus x^*/k is an ϵ-approximate fixed point of f. ◀

3 Computing a Tarski fixed point is in PLS ∩ PPAD

For a monotone function $f : [N]^d \to [N]^d$ (with respect to the coordinate-wise ordering), we are interested in computing a fixed point $x^* \in \text{Fix}(f)$, which we know exists by Tarski's theorem. We shall formally define this as a discrete total search problem, using a standard construction to avoid the "promise" that f is monotone.

Recall that a general discrete *total search problem* (with polynomially bounded outputs), Π, has a set of *valid input instances* $D_\Pi \subseteq \{0,1\}^*$, and associates with each valid input instance $I \in D_\Pi$, a non-empty set $\mathcal{O}_I \subseteq \{0,1\}^{p_\Pi(|I|)}$ of *acceptable outputs*, where $p_\Pi(\cdot)$ is some polynomial. (So the bit encoding length of every acceptable output is polynomially bounded in the bit encoding length of the input I.) We are interested in the complexity of the following total search problem:

▶ **Definition 3.** `Tarski`:

Input: *A function $f : [N]^d \to [N]^d$ with $N = 2^n$ for some $n \geq 1$, given by a boolean circuit, C_f, with $(d \cdot n)$ input gates and $(d \cdot n)$ output gates.*

Output: *Either a (any) fixed point $x^* \in \text{Fix}(f)$, or else a witness pair of vectors $x, y \in [N]^d$ such that $x \leq y$ and $f(x) \not\leq f(y)$.*

Note `Tarski` is a *total* search problem: If f is monotone, it will contain a fixed point in $[N]^d$, and otherwise it will contain such a witness pair of vectors that exhibit non-monotonicity. (If it is non-monotone it may of course have both witnesses for non-monotonicity and fixed points.)

Tarski ∈ PLS

Recall that a total search problem, Π, is in the complexity class PLS (*Polynomial Local Search*) if it satisfies all of the following conditions (see [16, 26]):

1. For each valid input instance $I \in D_\Pi \subseteq \{0,1\}^*$ of Π, there is an associated non-empty set $S_I \subseteq \{0,1\}^{p(|I|)}$ of *solutions*, and an associated *payoff function*[3], $g_I : S_I \to \mathbb{Z}$. For each $s \in S_I$, there is an associated set of *neighbors*, $\mathcal{N}_I(s) \subseteq S_I$.
   A solution $s \in S_I$ is called a *local optimum* (local maximum) if for all $s' \in \mathcal{N}_I(s)$, $g_I(s) \geq g_I(s')$. We let $\mathcal{O}_I$ denote the set of all local optima for instance I. (Clearly $\mathcal{O}_I$ is non-empty, because S_I is non-empty.)
2. There is a polynomial time algorithm, A_Π, that given a string $I \in \{0,1\}^*$, decides whether I is a valid input instance $I \in D_\Pi$, and if so outputs some solution $s_0 \in S_I$.
3. There is a polynomial time algorithm, B_Π, that given valid instance $I \in D_\Pi$ and a string $s \in \{0,1\}^{p(|I|)}$, decides whether $s \in S_I$, and if so, outputs the payoff $g_I(s)$.
4. There is a polynomial time algorithm, H_Π, that given valid instance $I \in D_\Pi$ and $s \in S_I$, decides whether s is a local optimum, i.e., whether $s \in \mathcal{O}_I$, and otherwise computes a strictly improving neighbor $s' \in \mathcal{N}_I(s)$, such that $g_I(s') > g_I(s)$.

▶ **Theorem 4.** Tarski ∈ PLS.

Proof Sketch. Each valid input instance $I_f \in D_{\texttt{Tarski}} \subseteq \{0,1\}^*$ of Tarski is an encoding of a function $f : [N]^d \to [N]^d$ via a boolean circuit C_f. We can view the problem Tarski as a polynomial local search problem, as follows:

Define the set of "solutions" associated with valid input I_f to be the disjoint union $S_{I_f} = S'_{I_f} \cup S''_{I_f}$, where $S'_{I_f} = \{x \in [N]^d \mid x \leq f(x)\}$ and $S''_{I_f} = \{(x,y) \in [N]^d \times [N]^d \mid x \leq y \wedge f(x) \not\leq f(y)\}$. Clearly, $\mathrm{Fix}(f) \subseteq S'_{I_f} \subseteq S_{I_f}$. Let the payoff function $g_{I_f} : S_{I_f} \to \mathbb{Z}$, be defined as follows. For $x \in S'_{I_f}$, $g_{I_f}(x) := \sum_{i=1}^d x_i$; for $(x,y) \in S''_{I_f}$, $g_{I_f}(x,y) := (dN)+1$. We define the neighbors of solutions as follows. For any $x \in S'_{I_f}$, if $f(x) \leq f(f(x))$ then let the neighbors of x be the singleton-set $\mathcal{N}_{I_f}(x) := \{f(x)\}$. Note that in this case again $f(x) \in S'_{I_f}$. Otherwise, if $f(x) \not\leq f(f(x))$, then let $\mathcal{N}_{I_f}(x) := \{(x, f(x))\}$. Note that in this case $(x, f(x)) \in S''_{I_f}$, since $f(x) \not\leq f(f(x))$. For $(x,y) \in S''_{I_f}$, let $\mathcal{N}_{I_f}(x,y) := \emptyset$ be the empty set. Thus, the set of local optima is by definition $\mathcal{O}_{I_f} = \{x \in S'_{I_f} \mid \sum_{i=1}^d x_i \geq \sum_{i=1}^d f_i(x)\} \cup S''_{I_f}$.

Observe that in fact $\mathcal{O}_{I_f} = \mathrm{Fix}(f) \cup S''_{I_f}$. Indeed, if $x \in \mathcal{O}_{I_f}$ then $x \in S'_{I_f}$ meaning $x \leq f(x)$, and also $\sum_{i=1}^d x_i \geq \sum_{i=1}^d f_i(x)$. But this is only possible if $f(x) = x$, i.e., $x \in \mathrm{Fix}(f)$. Likewise, if $(x,y) \in \mathcal{O}_{I_f}$ then $(x,y) \in S''_{I_f}$. On the other hand, if $x \in \mathrm{Fix}(f)$, then clearly $x \in S'_{I_f}$ and $\sum_{i=1}^d x_i = \sum_{i=1}^d f_i(x)$, hence $x \in \mathcal{O}_{I_f}$.

It is possible then to define polynomial time algorithms $A_{\texttt{Tarski}}$, $B_{\texttt{Tarski}}$ and $H_{\texttt{Tarski}}$, as required in conditions 2, 3, 4 in the definition of PLS; see the full paper for details. It follows that Tarski is in PLS. ◀

Tarski ∈ PPAD

To show that Tarski ∈ PPAD, we first show that Tarski $\in P^{\mathsf{PPAD}}$ meaning that the total search problem Tarski can be solved by a polynomial time algorithm, $\mathcal{M}$, with oracle access to PPAD. The algorithm $\mathcal{M}$ should take an input $I_f \in \{0,1\}^*$, and firstly decide whether it is a valid instance $I_f \in D_{\texttt{Tarski}}$, and if so it can make repeated, adaptive, calls to an

[3] Or, cost function, if we were considering local minimization. But here we focus on local maximization.

oracle for solving a PPAD total search problem. After at most polynomial time (and hence polynomially many such oracle calls) as a function of the input size $|I_f|$, $\mathcal{M}$ should output either an integer vector $x \in \mathrm{Fix}(f)$, or else output a pair of vectors $x, y \in [N]^d$ with $x \leq y$ and $f(x) \not\leq f(y)$, which witness non-monotonicity of the function $f : [N]^d \to [N]^d$ defined by the input instance I_f.

Once we have established that $\texttt{Tarski} \in P^{\mathsf{PPAD}}$, the fact that $\texttt{Tarski} \in \mathsf{PPAD}$ will follow as a simple corollary, using a prior result of Buss and Johnson [2], who showed that PPAD is closed under polynomial-time Turing reductions.

There are a number of equivalent ways to define the total search complexity class PPAD. Rather than give the original definition ([20]), we will use an equivalent characterization of PPAD (a.k.a., linear-FIXP) from [12]. Informally, according to this characterization, a discrete total search problem, Π, is in PPAD if and only if it can be reduced in P-time to computing a Brouwer fixed point of an associated "polynomial piecewise-linear" continuous function that maps a non-empty convex polytope to itself. That is, every instance I of Π can be associated with a polynomial piecewise-linear function F_I on a polytope $W(I)$, such that from any rational fixed point of F_I we can obtain in polynomial time an acceptable output for the instance I. By Brouwer's theorem, the set $\mathrm{Fix}(F_I) = \{x \in W(I) \mid F_I(x) = x\}$ of fixed points of F_I is non-empty. Moreover, because of the "polynomial piecewise-linear" nature of F_I, $\mathrm{Fix}(F_I)$ must also contain a *rational* fixed point x^*, with polynomial bit complexity as a function of $|I|$ (see [12], Theorem 5.2). See [12], section 5, for more details on this characterization of PPAD.

Given two vectors $l \leq h \in \mathbb{Z}^d$, let $L(l, h) = \{x \in \mathbb{Z}^d \mid l \leq x \leq h\}$, and let $B(l, h) = \{x \in \mathbb{R}^d \mid l \leq x \leq h\}$.

▶ **Theorem 5.** $\texttt{Tarski} \in P^{\mathsf{PPAD}}$.

Proof Sketch. Suppose we are given an instance $I_f \in D_{\texttt{Tarski}}$ of Tarski, corresponding to a function $f : [N]^d \to [N]^d$ (given by a boolean circuit C_f).

Let $a = \mathbf{1} \in \mathbb{Z}^d$, and $b = \mathbf{N} \in \mathbb{Z}^d$, denote the all 1, and all N, vectors respectively. We first extend the discrete function f to a (polynomial piecewise-linear) continuous function $f' : B(a, b) \to B(a, b)$, by a suitable linear interpolation. For this purpose we use a specific simplicial subdivision of $B(a, b)$, known as *Freudenthal's simplicial division* [15], which has a certain monotonicity property that is important for the proof. By Brouwer's theorem, f' has a fixed point in $B(a, b)$, and since it is polynomial piecewise-linear, finding a fixed point x^* is in PPAD. However, f' may have non-integer fixed points that do not correspond to (and are not close to) any fixed point of f (indeed, since we do not apriori know that f is monotone, there may not be any integer fixed points). Nevertheless, we show that finding any such fixed point x^* of f' allows us to make progress towards either finding a discrete fixed point of f (if it is monotone), or finding witnesses for a violation of monotonicity of f.

Specifically, we argue that there are two (integer) vertices $u \geq v$ of the simplex of the subdivision that contains x^* such that, if f is monotone, then $f(u) \geq u$ and $f(v) \leq v$ (in all coordinates). If $f(u) \not\geq u$, or $f(v) \not\leq v$, then f is not monotone, and we show that we can find a witness pair for the non-monotonicity of f.

Assume on the other hand that $f(u) \geq u$ and $f(v) \leq v$. Note that in that case, if f is monotone, then f maps the sublattice $L(u, b)$ to itself, and it also maps the disjoint sublattice $L(a, v)$ to itself. Thus, if f is monotone, f must have an integer fixed point in both $L(a, v)$ and $L(u, b)$.

So, we can choose the smaller of these two sublattices, consider the function f restricted to that sublattice, and continue recursively to find a fixed point in that sublattice (if f is monotone) or a violation of monotonicity. If f is not monotone, it is possible that it maps

some points in the sublattice $L(a, v)$ (or $L(u, b)$) to points outside. Therefore, in the recursive call for the sublattice, when we define the piecewise-linear function f' on the corresponding box $B(a, v)$ (or $B(u, b)$) we take the maximum with a and minimum with v (or u and b respectively), i.e., threshold it, so that it maps the box to itself, and hence it is a Brouwer function. When the PPAD oracle gives us back a fixed point for this (possibly thresholded) function f', we argue that if the thresholding mattered in this regard, then we can detect it and produce a violating pair to monotonicity.

Every iteration decreases the total number of points in our current lattice by a factor of 2, from the number of points in the original lattice $L(a, b)$. So after a polynomial number of iterations in $(d + \log N)$, we either find a fixed point of f, or we find a witness pair of integer vectors that witness the non-monotonicity of f. ◀

▶ **Corollary 6.** `Tarski` $\in$ PPAD.

Proof. This follows immediately from Theorem 5, combined with a result due to Buss and Johnson ([2], Theorem 6.1), who showed that PPAD is closed under polynomial-time Turing reductions. ◀

4 The 2-dimensional lower bound

Consider a monotone function defined on the $N \times N$ grid $f : [N]^2 \mapsto [N]^2$. Let A be any (randomized) *black-box algorithm* for finding a fixed point of the function by computing a sequence of queries of the form $f(x, y) =?$; A can of course be *adaptive* in that any query can depend in arbitrarily complex ways on the answers to the previous queries. For example, the divide-and-conquer algorithm described in the introduction is a black box algorithm. The following result suggests that this algorithm is optimal for two dimensions.

▶ **Theorem 7.** *Given black-box access to a monotone function $f : [N]^2 \to [N]^2$, any (randomized) algorithm for finding a fixed point of f requires $\Omega(\log^2 N)$ queries (in expectation).*

Below, we *sketch* a proof. For the full proof see [11]. The proof constructs a hard distribution of such functions.

The basic construction

Given a monotone path from $(1, 1)$ to (N, N) on the $N \times N$ grid graph and a point (i^*, j^*) on the path, we construct f as follows:

- We let (i^*, j^*) be the unique fixed point of f, i.e. $f(i^*, j^*) \triangleq (i^*, j^*)$.
- At all other points on the path, f is directed towards the fixed point. For a point (x, y) on the path that is dominated by (i^*, j^*), we let $f(x, y)$ be the next point on the path, i.e. $f(x, y) = (x + 1, y)$ or $f(x, y) = (x, y + 1)$. Similarly, for a point (x, y) that is on the path and dominates (i^*, j^*), we let $f(x, y)$ be the previous point on the path.
- For all points outside the path, f is directed towards the path. Observe that the path partitions $[N]^2$ into three (possibly empty) subsets: below the path, the path, and above the path. For a point (x, y) below the path, we set $f(x, y) \triangleq (x - 1, y + 1)$. Similarly, for a point (x, y) above the path, $f(x, y) \triangleq (x + 1, y - 1)$.

An example of such a function $f : [5]^2 \to [5]^2$ is given in Figure 1.

▷ Claim 8. For any choice of path and point (i^*, j^*) on the path, f constructed as above is monotone.

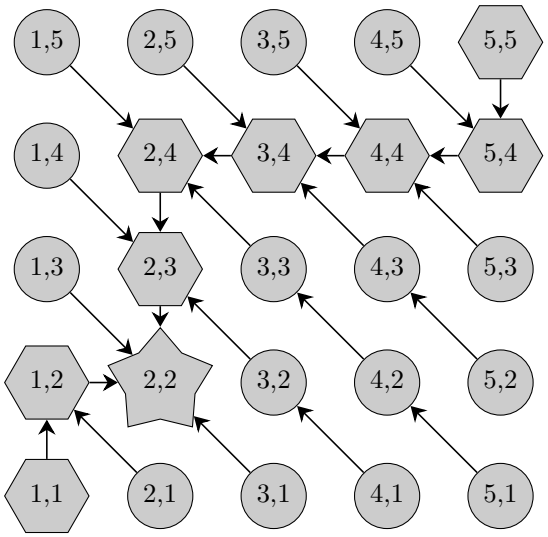

Figure 1 A 2-dimensional "*herringbone*" monotone function.

Choosing the fixed point

In our hard distribution, once we fix a path, we choose (i^*, j^*) uniformly at random among all points on the path.

▷ Claim 9. Given oracle access to f and the path, any (randomized) algorithm that finds a point (i', j') on the path that is within $\sqrt{N}$ (Manhattan distance) from (i^*, j^*) requires (in expectation) querying of f at $\Omega(\log N)$ points on the path that are pairwise at least $\sqrt{N}$ apart.

Proof. Observe that once we fix the path, the values of f outside the path do not reveal information about the location of (i^*, j^*). The lower bound now follows from the standard lower bound for binary search. ◁

Choosing the central path

Our goal now is to prove that it is hard to find many distant points on the path. To simplify the analysis, we will only consider the special case where all points (x, y) on the path satisfy $x - y \in \left[-N^{1/4}, N^{1/4}\right]$. We partition the $N \times N$ grid into $\Theta\left(\sqrt{N}\right)$ regions of the form $R_a \triangleq \left\{(x, y) \mid x + y \in [a, a + \sqrt{N})\right\}$. Notice that each region intersects the path at exactly $\sqrt{N}$ points. The path enters each region[4] at a point (x, y) for a value $x - y$ chosen uniformly at random among $\left[-N^{1/4}, N^{1/4}\right]$. We will argue (Lemma 12 below) that in order to find a point on the path in any region R_a, the algorithm must query the function at $\Omega(\log N)$ points in R_a or its neighboring regions.

Each region is further partitioned into $\Theta\left(N^{1/4}\right)$ sub-regions $S_a \triangleq \left\{(x, y) \mid x + y \in [a, a + 2N^{1/4})\right\}$. For each region, we choose a special sub-region uniformly at random. In all non-special sub-regions, the path proceeds while maintaining

[4] For the first and last region, the path is obviously forced to start at $(1, 1)$ (respectively end at (N, N)); but those two regions can only account for two of the $\Omega(\log N)$ distant path points required by Claim 9, so we can safely ignore them.

$x-y$ fixed, up to ± 1. Inside the special sub-region, the value of $x-y$ for path points changes from the value chosen at random for the current region, to the value chosen at random for the next region.

Given a choice of random $x-y$ entry point for each region, and a random special sub-region for each region, we consider an arbitrary path that satisfies the description above. This completes the description of the construction.

▷ Claim 10. Finding (i.e., querying any point in) the special sub-region in region R_a requires $\Omega(\log N)$ queries (in expectation) to points in R_a.

Let S_a and S_b be the special sub-regions of two consecutive regions. Let $T \triangleq \{(x,y) \mid x+y \in [a+2N^{1/4}, b)\}$ be the union of all the sub-regions between S_a and S_b. Observe that the value of $x-y$ remains fixed (up to ± 1) for all points in the intersection of the path with T. Also, the construction of f outside $S_a \cup T \cup S_b$ does not depend at all on this value.

▷ Claim 11. If the algorithm does not query any point in $S_a \cup S_b$, then in order to find (i.e., query) any point in the intersection of the path and T, the algorithm must query (in expectation) $\Omega(\log N)$ points from T.

By Claim 10, finding S_a or S_b requires at least $\Omega(\log N)$ queries to the regions containing them. Therefore, the above two claims together imply:

▶ **Lemma 12.** *In order to query a point in the intersection of the path and region R_a, any algorithm must query at least $\Omega(\log N)$ points (in expectation) in R_a or its neighboring regions.*

Therefore, in order to find $\Omega(\log N)$ points on the path that are pairwise at least $\sqrt{N}$ apart, the algorithm must make a total of $\Omega\left(\log^2 N\right)$ queries (in expectation), completing the proof of Theorem 7.

An alternative proof

In the full version of this paper (see [11]) we provide an alternative proof, showing that any *deterministic* black box algorithm requires $\Omega(\log^2 N)$ oracle queries to find a Tarski fixed point of a monotone function $f : [N]^2 \to [N]^2$ given by an oracle.

▶ **Theorem 13.** *Any deterministic black box algorithm for finding a Tarski fixed point in two dimensions needs $\Omega(\log^2 N)$ queries.*

The alternative proof appears to be more promising for generalization to higher dimensions. However, the underlying monotone functions $f : [N]^2 \to [N]^2$ on which the alternative lower bound is established are again the "*herringbone*" functions used in the proof of Theorem 7. The alternative proof uses a potential argument, and its gist amounts to showing that any such algorithm must solve $\Omega(\log N)$ *independent* one-dimensional problems.

5 Supermodular Games

A brief intro to supermodular games

A *supermodular game* is a game in which the set S_i of strategies of each player i is a complete lattice, and the utility (payoff) functions u_i satisfy certain conditions. Let k be the number of players and let $S = \Pi_{i=1}^{k} S_i$ be the set of strategy profiles. As usual, we use s_i to denote a strategy for player i and s_{-i} to denote a tuple of strategies for the other players. The conditions on the utility functions u_i are the following:

C1. $u_i(s_i, s_{-i})$ is upper semicontinuous in s_i for fixed s_{-i}, and it is continuous in s_{-i} for each fixed s_i, and has a finite upper bound.

C2. $u_i(s_i, s_{-i})$ is supermodular in s_i for fixed s_{-i}.

C3. $u_i(s_i, s_{-i})$ has increasing differences in s_i and s_{-i}.

A function $f : L \to \mathbb{R}$ is *supermodular* if for all $x, y \in L$, it holds $f(x) + f(y) \leq f(x \wedge y) + f(x \vee y)$. A function $f : L_1 \times L_2 \to \mathbb{R}$, where L_1, L_2 are lattices, has *increasing differences* in its two arguments, if for all $x' \geq x$ in L_1 and all $y' \geq y$ in L_2, it holds that $f(x', y') - f(x, y') \geq f(x', y) - f(x, y)$.

The broader class of *games with strategic complementarities* (GSC) relaxes somewhat the conditions C2 and C3 into C2', C3' which depend only on ordinal information on the utility functions, i.e. how the utilities compare to each other rather than their precise numerical values. The supermodularity requirement of C2 is relaxed to quasi-supermodularity, where a function $f : L \to \mathbb{R}$ is *quasi-supermodular* if for all $x, y \in L$, $f(x) \geq f(x \wedge y)$ implies $f(y) \leq f(x \vee y)$, and if the first inequality is strict, then so is the second. The increasing differences requirement of C3 is relaxed to the *single-crossing condition*, where a function $f : L_1 \times L_2 \to \mathbb{R}$, satisfies the single crossing condition, if for all $x' > x$ in L_1 and all $y' > y$ in L_2, it holds that $f(x', y) \geq f(x, y)$ implies $f(x', y') \geq f(x, y')$, and if the first inequality is strict then so is the second. All the structural and algorithmic properties below of supermodular games hold also for games with strategic complementarities.

We will consider here games where each S_i is a discrete (or continuous) finite box in d_i dimensions of size N in each coordinate. We let $d = \sum_{i=1}^{k} d_i$ be the total number of coordinates. In the discrete case, condition C1 is trivial. Condition C2 is trivial if $d_i = 1$ (all functions in one dimension are supermodular), but nontrivial for 2 or more dimensions. C3 is nontrivial.

Supermodular games (and GSC) have pure Nash equilibria. Furthermore, the pure Nash equilibria form a complete lattice [17], thus there is a highest and a lowest equilibrium. Another important property is that the best response correspondence $\beta_i(s_{-i})$ for each player i has the property that (1) both $\sup \beta_i(s_{-i})$ and $\inf \beta_i(s_{-i})$ are in $\beta_i(s_{-i})$, and (2) both functions $\sup \beta_i(\cdot)$ and $\inf \beta_i(\cdot)$ are monotone functions [24]. The function $\bar{\beta}(s) = (\sup \beta_1(s_{-1}), \ldots, \sup \beta_k(s_{-k}))$ of the supremum best responses is a monotone function from S to itself, and its greatest fixed point is the highest Nash equilibrium of the game. The function $\underline{\beta}(s) = (\inf \beta_1(s_{-1}), \ldots, \inf \beta_k(s_{-k}))$ of the infimum best responses is also a monotone function, and its least fixed point is the lowest Nash equilibrium of the game.

Complexity of equilibrium computation in supermodular games

Given a supermodular game, the relevant problems include: (a) find a Nash equilibrium (anyone)[5], and (b) find the highest or the lowest equilibrium. In the case of continuous domains, we again have to relax to an approximate solution. We assume that we have access to a best response function, e.g. $\bar{\beta}(\cdot)$ and/or $\underline{\beta}(\cdot)$, as an oracle or as a polynomial-time function. The monotonicity of these functions implies then easily the following:

[5] Whenever we speak of finding a Nash Equilibrium (NE) for a supermodular game, we mean a *pure* NE, as we know that these exists.

▶ **Proposition 14.**

1. *The problem of computing a pure Nash equilibrium of a k-player supermodular game over a discrete finite strategy space $\Pi_{i=1}^{k}[N]^{d_i}$ reduces to the problem of computing a fixed point of a monotone function over $[N]^d$ where $d = \sum_{i=1}^{k} d_i$. Computing the highest (or lowest) Nash equilibrium reduces to computing the greatest (or lowest) fixed point of a monotone function.*
2. *For games with continuous box strategy spaces, $\Pi_{i=1}^{k}[1, N]_i^d$, and Lipschitz continuous utility functions with Lipschitz constant K, the problem of computing an ϵ-approximate Nash equilibrium reduces to exact fixed point computation point for a monotone function with a discrete finite domain $[NK/\epsilon]^d$.*

Proof.

1. Follows from the monotonicity of $\bar{\beta}(\cdot)$ and $\underline{\beta}(\cdot)$. If s is fixed point of $\bar{\beta}(\cdot)$, then $s_i = \sup \beta_i(s_{-i})$ is a best response to s_{-i} for all i (since $\sup \beta_i(s_{-i}) \in \beta_i(s_{-i})$), therefore s is a Nash equilibrium of the game. The GFP of $\bar{\beta}(\cdot)$ is the highest Nash equilibrium. Similarly, every fixed point of $\underline{\beta}(\cdot)$ is an equilibrium of the game, and the LFP of $\underline{\beta}(\cdot)$ is the lowest equilibrium.
2. Suppose that the utility functions are Lipschitz continuous with Lipschitz constant K. To compute an ϵ-approximate Nash equilibrium of the game, it suffices to find a ϵ/K-approximate fixed point of the function $\bar{\beta}(\cdot)$. For, if s is such an approximate fixed point and $s' = \bar{\beta}(s)$, then $|s' - s| \leq \epsilon/K$ in every coordinate. Hence $|u_i(s_i, s_{-i}) - u_i(s_i', s_{-i})| \leq \epsilon$, and s_i' is a best response to s_{-i}, hence s is an ϵ-approximate equilibrium. Computing an ϵ/K-approximate fixed point of the function $\bar{\beta}(\cdot)$ on the continuous domain, reduces by Proposition 2 to the exact fixed point problem for the discrete domain $[NK/\epsilon]^d$. ◀

Not every monotone function can be the (sup or inf) best response function of a game. In particular, a best response function has the property that the output values for the components corresponding to a player depend only on the input values for the other components corresponding to the other players. Thus, for example, for two one-dimensional players, if the function $f(x, y)$ is the best response function of a game, it must satisfy $f_1(x, y) = f_1(x', y)$ for all x, x', y, and $f_2(x, y) = f_2(x, y')$ for all x, y, y'. This property helps somewhat in improving the time needed to find a fixed point, and thus an equilibrium of the game, as noted below. For example, in the case of two one-dimensional players, an equilibrium can be computed in $O(\log N)$ time, instead of the $\Omega(\log^2 N)$ time needed to find a fixed point of a general monotone function in two dimensions.

▶ **Theorem 15.** *Given a supermodular game with two players with discrete strategy spaces $[N]^{d_i}$, $i = 1, 2$ with access to the sup (or inf) best response function $\bar{\beta}(\cdot)$ (or $\underline{\beta}(\cdot)$), we can compute an equilibrium in time $O((\log N)^{\min(d_1, d_2)})$. More generally, for k players with dimensions $d_1, \ldots, d_k$, an equilibrium can be computed in time $O((\log N)^{d'})$, where $d' = \sum_i d_i - \max_i d_i$.*

Proof. Suppose that we have access to the sup best response $\bar{\beta}(\cdot)$. Assume without loss of generality that the first player has the maximum dimension, $d_1 = \max_i d_i$. We apply the divide-and-conquer algorithm, but take advantage of the property of the monotone function $\bar{\beta}$ that the first d_1 components of $\bar{\beta}(x)$ do not depend on the first d_1 coordinates of x. As a consequence, for any fixed assignment to the other coordinates, i.e. choice of a strategy profile s_{-1} for all the players except the first player, the induced function on the first d_1 coordinates maps every point to the best response $\bar{\beta}_1(s_{-1})$ of player 1. Thus the fixed point of the induced function is simply $\bar{\beta}_1(s_{-1})$, it can be computed with one call to $\bar{\beta}$, and there is no need to recurse on the first d_1 coordinates. It follows that the algorithm takes time at most $O((\log N)^{d'})$, where $d' = \sum_i d_i - \max_i d_i$. ◀

Conversely, we can reduce the fixed point computation problem for an arbitrary monotone function to the equilibrium computation problem for a supermodular game with two players.

▶ **Theorem 16.**

1. *Given a monotone function f on $[N]^d$ (resp. $[1,N]^d$) we can construct a supermodular game G with two players, each with strategy space $[N]^d$ (resp. $[1,N]^d$), so that the pure Nash equilibria of G correspond to the fixed points of f.*
2. *More generally, the fixed point problem for a monotone function f in d dimensions can be reduced to the pure Nash equilibrium problem for a supermodular game with any number $k \geq 2$ of players with any dimensions $d_1, \ldots, d_k$, provided that $\sum_i d_i \geq 2d$ and $\sum_i d_i - \max_i d_i \geq d$.*

Proof Sketch.

1. We will define the utility functions u_i so that the best responses β_i of both players are functions (i.e. are unique). For player 1, the best response will be $\beta_1(y) = y$, for all $y \in [N]^d$, and for player 2, the best response will be $\beta_2(x) = f(x)$, for all $x \in [N]^d$. If x is a fixed point of f, then (x,x) is an equilibrium of the game, since $\beta(x,x) = (x, f(x)) = (x,x)$. Conversely, if (x,y) is an equilibrium of the game, then $\beta(x,y) = (x,y)$, therefore $x = y$ and $y = f(x)$, hence $x = f(x)$. Thus, the set of equilibria of G is $\{(x,x) | x \in Fix(f)\}$.
   The utility function for player 1 is set to $u_1(x,y) = -(x-y)^2 = -\sum_{j=1}^{d}(x_j - y_j)^2$. The utility function for player 2 is $u_2(x,y) = -(f(x)-y)^2 = -\sum_{j=1}^{d}(f_j(x) - y_j)^2$. Obviously, the best response functions are as stated above, $\beta_1(y) = y$ and $\beta_2(x) = f(x)$.
   It can be verified that the utility functions u_1, u_2 satisfy conditions C2 and C3 in the definition of supermodular games (C2 with equality actually).
2. Order the players in increasing order of their dimension, let T be the ordering of all the $\sum_{i=1}^{k} d_i$ coordinates consisting first of the set $Co(1)$ of coordinates of player 1 (in any order), then the set $Co(2)$ of coordinates of player 2, and so forth. Number the coordinates in the order T from 1 to $\sum_{i=1}^{k} d_i$, and label them cyclically with the labels $1, \ldots, d$.
   We define the (unique) best response function β as follows. For every coordinate $j \leq d$ (in the ordering T), we set $\beta_j(x) = f_j(x')$, where x' is a subvector of x with d coordinates that have distinct labels $1, \ldots, d$ and which belong to different players than coordinate j. The subvector x' is defined as follows. Suppose that coordinate j belongs to player r ($j \in Co(r)$), and let $t = \sum_{i=1}^{r-1} d_i$. If $d_r \leq d$, then x' is the subvector of x that consists of the first t coordinates (in the order T) and the coordinates $t+1+d, \ldots, 2d$; note that all these coordinates do not belong to player r. If $d_r > d$, then $r < k$ (since $\sum_i d_i - \max_i d_i \geq d$). In this case, let x' be the subvector of x consisting of the last d coordinates (in T); all of these belong to player $k \neq r$. For coordinates $j > d$, we set $\beta_j(x) = x_{j'}$, where $j' \in [d]$ is equal to $j \mod d$, unless j' belongs to the same player r as j, in which case $d_r > d$, hence $r \neq k$; in this case we set $\beta_j(x) = x_{j"}$ for some (any) coordinate $j"$ of the last player k that is labeled j'.
   We define the utility functions of the players so that they yield the above best response function β. Namely, we define the utility function of player i to be $u_i(x) = -\sum_{j \in Co(i)}(x_j - \beta_j(x))^2$. It can be verified as in part 1 that the utility functions satisfy conditions C2 and C3. It can be easily seen also that at any equilibrium of the game, all coordinates with the same label must have the same value, and the corresponding d-vector x is a fixed point of f. Conversely, for any fixed point x of f, the corresponding strategy profile of the game is an equilibrium. ◀

Since the 2-dimensional monotone fixed point problem requires $\Omega(\log^2 N)$ queries by Theorem 7, it follows that the equilibrium problem for two 2-dimensional players also requires $\Omega(\log^2 N)$ queries, which is tight because it can be also solved in $O(\log^2 N)$ time by Theorem 15. Similarly, for higher dimensions d, if the monotone fixed point problem requires $\Omega(\log^d N)$ queries then the equilibrium problem for two d-dimensional players is also $\Theta(\log^d N)$.

The same reduction from monotone functions to supermodular games of Theorem 16, combined with Proposition 1 implies the hardness of computing the highest and lowest equilibrium.

▶ **Corollary 17.** *It is NP-hard to compute the highest and lowest equilibrium of a supermodular game with two 1-dimensional players with explicitly given polynomial-time best response (and utility) functions.*

6 Condon's and Shapley's stochastic games reduce to `Tarski`

In this section, we show that computing the exact (rational) value of Condon's simple stochastic games ([4]), as well as computing the (irrational) value of Shapley's more general (stopping/discounted) stochastic games [21] to within a given desired error $\epsilon > 0$ (given in binary), is polynomial time reducible to `Tarski`.

A *simple stochastic game*[6] (SSG) is a 2-player zero-sum game, played on the vertices of an edge-labeled directed graph, specified by $G = (V, V_0, V_1, V_2, \delta)$, whose vertices $V = \{v_1, \ldots, v_n\}$ include two special sink vertices, a **0**-sink, v_{n-1}, and a **1**-sink, v_n, and where the rest of the vertices $V \setminus \{v_{n-1}, v_n\} = \{v_1, \ldots, v_{n-2}\}$ are partitioned into three disjoint sets V_0 (random), V_1 (max), and V_2 (min). The labeled directed edge relation is $\delta \subseteq (V \setminus \{v_{n-1}, v_n\}) \times ((0, 1] \cup \bot) \times V$. For each "random" node $u \in V_0$, every outgoing edge $(u, p_{u,v}, v) \in \delta$ is labeled by a positive probability $p_{u,v} \in (0, 1]$, such that these probabilities sum to 1, i.e., $\sum_{\{v \in V | (u,p_{u,v},v) \in \delta\}} p_{u,v} = 1$. We assume, for computational purposes, that the probabilities $p_{u,v}$ are rational numbers (given as part of the input, with numerator and denominator given in binary). The outgoing edges from "max" (V_1) and "min" (V_2) nodes have an empty label, "$\bot$". We assume each vertex $u \in V \setminus \{v_{n-1}, v_n\}$ has at least one outgoing edge. Thus in particular, for any node $u \in V_1 \cup V_2$ there exists an outgoing edge $(u, \bot, v) \in \delta$ for some $v \in V$. Finally, there is a designated start vertex $s \in V$.

A play of the game transpires as follows: a token is initially placed on s, the start node. Thereafter, during each "turn", when the token is currently on a node $u \in V$, unless u is already a sink node (in which case the game halts), the token is moved across an outgoing edge of u to the next node by whoever "controls" u. For a random node $u \in V_0$, which is controlled by "nature", the outgoing edge is chosen randomly according to the probabilities $(p_{u,v})_{v \in V}$. For $u \in V_1$, the outgoing edge is chosen by player 1, the max player, who aims to maximize the probability that the token will eventually reach the **1**-sink. For $u \in V_2$, the outgoing edge is chosen by player 2, the min player, who aims to minimize the probability that the token will eventually reach the **1**-sink. The game halts if the token ever reaches either of the two sink nodes.

[6] The definition we give here for SSGs is slightly more general than Condon's original definition in [4]. Specifically, Condon allows edge probabilities of 1/2 only, and also assumed that the game is a "stopping game", meaning it halts with probability 1, regardless of the strategies of the two players. It is well known that our more general definition does not alter the difficulty of computing the game value and optimal strategies: solving general SSGs can be reduced in P-time to solving SSGs in Condon's more restricted form.

For every possible start node $s = v_i \in V$, this zero-sum game has a well defined *value*, $q_i^* \in [0, 1]$. This is, by definition, a probability such that player 1, the max player (and, respectively, player 2, the min player) has a strategy to "force" reaching the 1-sink with probability at least (respectively, at most) q_i^*, irrespective of what the other player's strategy is. In other words, these games are *determined*. Moreover q_i^* is a rational value whose encoding size, with numerator and denominator in binary, is polynomial in the bit encoding size of the SSG ([4]). Furthermore, both players have deterministic, memoryless (a.k.a., pure, positional) optimal strategies in the game (which do not depend on the specific start node s), in which for each vertex $u \in V_1$ (or $u \in V_2$) the max player (respectively the min player) chooses the same specific outgoing edge every time the token visits vertex u, regardless of the prior history of play prior to that visit to u.

Given an SSG, the goal is to compute the value of the game (starting at each vertex). Condon ([4]) already showed that the problem of deciding whether the value is $> 1/2$ is in NP $\cap$ co-NP, and it is a long-standing open problem whether this is in P-time. Moreover, the search problem of computing the value for an SSG is known to be in both PLS and PPAD (see [25] and [12]). In the full paper we show the following:

▶ **Proposition 18.** *The following total search problem is polynomial-time reducible to* `Tarski`*: Given an instance G of Condon's simple stochastic game, and given a start vertex $s = v_i \in V$, compute the exact (rational) value q_i^* of the game.*

Shapley's original stochastic games are more general, and involve simultaneous (independent) choices by the two players at each state (they are thus imperfect information games). The value of the game is in general irrational (even when all the input data is rational). We show that approximating the value of a Shapley game to within any given desired accuracy, $\epsilon > 0$ (given in binary as part of the input), is polynomial time reducible to `Tarski`.

Shapley's games are a class of two-player zero-sum "stopping", or equivalently "discounted", stochastic games. An instance of Shapley's stochastic game is given by $G = (V, A, P, s)$, where $V = \{v_1, \ldots, v_n\}$ is a set of n vertices (or "states"). $A = (A^1, A^2, \ldots, A^n)$ is a n-tuple of matrices, where, for each vertex, $v_i \in V$, A^i is an associated $m_i \times n_i$ *reward matrix*, where m_i and n_i are positive integers denoting, respectively, the number of distinct "actions" available to player 1 (the maximizer) and player 2 (the minimizer) at vertex v_i, and where for each pair of such actions, $j \in [m_i]$ and $k \in [n_i]$, $A^i_{j,k} \in \mathbb{Q}$ is a reward for player 1 (which we assume, for computational purposes, is a rational number given as input by giving its numerator and denominator in binary). Furthermore, for each vertex $v_i \in V$, and each pair of actions $j \in [m_i]$ and $k \in [n_i]$, $P^i_{j,k} \in [0, 1]^n$ is a vector of probabilities on the vertices V, such that $0 \leq P^i_{j,k}(r)$, and $\sum_{r=1}^n P^i_{j,k}(r) < 1$, i.e., the probabilities sum to *strictly less than* 1. Again, we assume each such probability $P^i_{j,k}(r) \in \mathbb{Q}$ is a rational number given as input in binary. Finally, the game specifies a designated start vertex $s \in V$.

A play of Shapley's game transpires as follows: a token is initially placed on s, the start node. Thereafter, during each "round" of play, if the token is currently on some node $v_i \in V$, both players simultaneously and independently choose respective actions $j \in [m_i]$ and $k \in [n_i]$, and player 1 then receives the corresponding reward $A^i_{j,k}$ from player 2; thereafter, for each $r \in [n]$ with probability $P^i_{j,k}(r)$ the token is moved from node v_i to node v_r, and with the remaining positive probability $q^i_{j,k} = 1 - \sum_{r=1}^n P^i_{j,k}(r) > 0$, the game "halts". Let $q = \min\{q^i_{j,k} \mid i, j, k\} > 0$ be the minimum such halting probability at any state, and under any pair of actions. Since q is positive, i.e., since there is positive probability $\geq q > 0$ of halting after each round, a play of the game eventually halts with probability 1. The goal of player 1 (player 2) is to maximize (minimize, respectively) the expected total reward that player 1 receives from player 2 during the entire play. A *strategy* for each player

specifies, based in principle on the entire history of play thusfar, a probability distribution on the actions available at the current token location. Given strategies σ_1 and σ_2 for player 1 and 2, respectively, let $r_i(\sigma_1, \sigma_2)$ denote the expected total payoff to player 1, starting at node $s = v_i \in V$. Shapley [21] showed that these games have a *value*, meaning that $\sup_{\sigma_1} \inf_{\sigma_2} r_i(\sigma_1, \sigma_2) = \inf_{\sigma_2} \sup_{\sigma_1} r_i(\sigma_1, \sigma_2)$. In fact, Shapley showed that both players have optimal stationary (but randomized) strategies in such games, i.e., optimal strategies that only depend on the current node where the token is located, not the prior history of play, but where players can randomize on their choice of actions at each node.

Let $r_i^* = \sup_{\sigma_1} \inf_{\sigma_2} r_i(\sigma_1, \sigma_2)$ denote the game value starting at vertex $s = v_i \in V$.[7] We show the following in the full paper.

▶ **Proposition 19.** *The following total search problem is polynomial-time reducible to* `Tarski`*: Given an instance G of Shapley's stochastic game, and given $\epsilon > 0$ (in binary), compute a vector $r' \in \mathbb{Q}^n$ such that $\|r^* - r'\|_\infty < \epsilon$.*

7 Conclusions and Discussion

We have studied the complexity of computing a Tarski fixed point for a monotone function over a finite discrete Euclidean grid, and we have shown that this problem essentially captures the complexity of computing a (ϵ-approximate) pure Nash equilibrium of a supermodular game. We have also shown that computing the value of Condon's and Shapley's stochastic games reduces to this `Tarski` fixed point problem, where the monotone function is given succinctly (by a boolean circuit).

We have provided several upper bounds for the `Tarski` problem, showing that it is contained in both PLS and PPAD. On the other hand, in the oracle model, for 2-dimensional monotone functions $f : [N]^2 \to [N]^2$, we have shown a $\Omega(\log^2 N)$ lower bound for the (expected) number of (randomized) queries required to find a Tarski fixed point, which matches the $O(\log^d N)$ upper bound for $d = 2$.

A key question left open by our work is to improve the lower bounds in the oracle model to higher dimensions. It is tempting to conjecture that for small (fixed) dimension d, a lower bound close to $\Omega(\log^d N)$ holds. On the other hand, we know that this cannot hold for arbitrary d and N, because we also have the dN upper bound, which is better than $\log^d N$ when $d = \omega(\frac{\log N}{\log \log N})$.

Another interesting open question is the relationship between the `Tarski` problem and the total search complexity class CLS [9], as well as the closely related recently defined class EOPL (which stands for "End of Potential Line" [13, 14]). EOPL is contained in CLS, which is contained in both PLS and PPAD. Is `Tarski` in CLS (or in EOPL)? That would be remarkable, as the proof that it is in PPAD is currently quite indirect. Conversely, can `Tarski` be proved to be CLS-hard (EOPL-hard)? (Recall from the previous section that some key problems in CLS related to stochastic games do reduce to `Tarski`.)

Another question worth considering is the complexity of the `unique-Tarski` problem, where the monotone function is further assumed (promised) to have a *unique* fixed point. Is `unique-Tarski` easier than `Tarski`? Note that our $\Omega(\log^2 N)$ lower bound in the oracle model, in dimension $d = 2$, applies on the family of "herringbone" functions which do have a unique fixed point.

[7] Note that we could also define r_i^* as $r_i^* = \max_{\sigma_1} \min_{\sigma_2} r_i(\sigma_1, \sigma_2)$, due to the existence of optimal strategies.

Finally, in this paper we have studied the Tarski fixed point problem only in the setting of monotone functions on the Euclidean grid $[N]^d$. Let us remark, however, that it is possible to consider a more general black box model, for monotone functions $f : L \to L$, over an arbitrary finite lattice $(L, \preceq)$, where the lattice's elements $L \subseteq \{0,1\}^n$ are encoded as binary strings of some given length n, and where we assume the entire lattice $(L, \preceq)$ is known *explicitly* by the querier, who moreover has unbounded computational power, but who only has oracle access to the monotone function $f : L \to L$. In the full version of this paper ([11]), generalizing the $\log^d N$ algorithm for Euclidean grids, we show that in this black box model there is a deterministic algorithm that computes a fixed point of $f : L \to L$ using $O(\log^d(|L|))$ queries to the function f, where d is the *dimension* of the lattice $(L, \preceq)$. The *dimension* of a lattice $(L, \preceq)$, and more generally the dimension of any partial order, can be defined as the smallest integer $d \geq 1$ such that the relation $\preceq$ is the intersection of d total orders on the same underlying set L. Equivalently, it is the smallest $d \geq 1$ such that there is an injective embedding of $(L, \preceq)$ in the Euclidean grid $([|L|]^d, \leq)$, where $\leq$ is the standard coordinate-wise partial order on $[|L|]^d$. Note that a lower bound of $\Omega(\log^2(|L|))$ queries for computing a fixed point of a monotone function $f : L \to L$ in this black box model follows directly from our lower bound of $\Omega(\log^2 N)$ for monotone functions on the 2D grid $f : [N]^2 \to [N]^2$. At present, we do not know any better lower bound than $\Omega(\log^2(|L|))$ in this black box model for arbitrary finite lattices.[8]

References

1 K. J. Arrow and G. Debreu. Existence of an equilibrium for a competitive economy. *Econometrica*, pages 265–290, 1954.

2 S. R. Buss and A. S. Johnson. Propositional proofs and reductions between NP search problems. *Annals of Pure and Applied Logic*, 163:1163–1182, 2012.

3 C.-L. Chang, Y.-D. Lyuu, and Y.-W. Ti. The complexity of Tarski's fixed point theorem. *Theoretical Computer Science*, 401:228–235, 2008.

4 A. Condon. The complexity of stochastic games. *Information and Computation*, 96:203–224, 1992.

5 C. Dang, Q. Qi, and Y. Ye. Computational models and complexities of Tarski's fixed points. Technical report, Stanford University, 2012.

6 C. Dang and Y. Ye. On the complexity of a class of discrete fixed point problems under the lexicographic ordering. Technical Report CY2018-3, City University of Hong Kong, 2018.

7 C. Dang and Y. Ye. On the complexity of an expanded Tarski's fixed point problem under the componentwise ordering. *Theoretical Computer Science*, 732:26–45, 2018.

8 C. Dang and Y. Ye. Personal commuication with the authors, March 2019.

9 C. Daskalakis and C. H. Papadimitriou. Continuous Local Search. In *Proceedings of 22nd ACM-SIAM Symp. on Discrete Algorithms (SODA'11)*, pages 790–804, 2011.

10 F. Echenique. Finding all equilibria in games with strategic complements. *Journal of Economic Theory*, 135(1):514–532, 2007.

11 K. Etessami, C. Papadimitriou, A. Rubinstein, and M. Yannakakis. Tarski's theorem, supermodular games, and the complexity of equilibria. arXiv preprint, 2019. `arXiv:1909.03210`.

12 K. Etessami and M. Yannakakis. On the complexity of Nash equilibria and other fixed points. *SIAM Journal on Computing*, 39(6):2531–2597, 2010.

[8] Note that this black box model is very different from the one considered in [3], where the lattice itself is not known explicitly, but is only accessible via an oracle for its partial order. Hence the linear $\Omega(|L|)$ lower bound on the number of queries (including queries to the partial order itself) given in [3] for finding a fixed point has no bearing on the black box model described here, where the lattice itself is explicitly known, and only the monotone function is given by an oracle.

13 J. Fearnley, S. Gordon, R. Mehta, and R. Savani. End of Potential Line. arXiv preprint, 2018. `arXiv:1804.03450`.

14 J. Fearnley, S. Gordon, R. Mehta, and R. Savani. Unique End of Potential Line. In *Proc. of 46th Int. Coll. on Automata, Languages, and Programming (ICALP'19)*, page 15 pages, 2019. (Full preprint: `arXiv:1811.03841`, 90 pages).

15 H. Freudenthal. Simplizialzerlegungen von beschrankter flachheit. *Annals of Mathematics*, 43:580–582, 1942.

16 D. S. Johnson, C. H. Papadimitriou, and M. Yannakakis. How easy is local search? *Journal of Computer and System Sciences*, 37:79–100, 1988.

17 P. Milgrom and J. Roberts. Rationalizability, learning, and equilibrium in games with strategic complementarities. *Econometrica*, pages 1255–1277, 1990.

18 J. Nash. Non-cooperative Games. *Annals of Mathematics*, pages 286–295, 1951.

19 N. Nisan, T. Roughgarden, E. Tardos, and V. V. Vazirani, editors. *Algorithmic Game Theory*. Cambridge University Press, 2007.

20 C. H. Papadimitriou. On the Complexity of the Parity Argument and Other Inefficient Proofs of Existence. *Journal of Computer and System Sciences*, 48(3):498–532, 1994.

21 L. S. Shapley. Stochastic Games. *Proceeding of the National Academy of Sciences*, 39(10):1095–1100, 1953.

22 A. Tarski. A lattice-theoretical fixpoint theorem and its applications. *Pacific Journal of Mathematics*, 5(2):285–309, 1955.

23 D. M. Topkis. Equilibrium Points in Nonzero-Sum n-Person Submodular Games. *SIAM Journal on control and optimization*, 17(6):773–787, 1979.

24 D. M. Topkis. *Supermodularity and Complementarity*. Princeton University Press, 2011.

25 M. Yannakakis. The analysis of local search problems and their heuristics. In *Proc. of Symp. on Theoretical Aspects of Computer Science (STACS'90)*, Springer LNCS 415, pages 298–311, 1990.

26 M. Yannakakis. Chapter 2: Computational Complexity. In E. Aarts and J. K. Lenstra, editors, *Local Search in Combinatorial Optimization*. John Wiley & Sons, 1997.