Tarski Lower Bounds from Multi-Dimensional Herringbones*

Simina Brânzei
Purdue University, West Lafayette, IN, USA

Reed C. Phillips
Purdue University, West Lafayette, IN, USA

Nicholas J. Recker
Epic, Verona, WI, USA

Abstract

Tarski's theorem states that every monotone function from a complete lattice to itself has a fixed point. We analyze the query complexity of finding such a fixed point on the k-dimensional grid of side length n under the $\leq$ relation. In this setting, there is an unknown monotone function $f : \{0, 1, \ldots, n-1\}^k \to \{0, 1, \ldots, n-1\}^k$ and an algorithm must query a vertex v to learn $f(v)$. The goal is to find a fixed point of f using as few oracle queries as possible.

We show that the randomized query complexity of this problem is $\Omega\left(\frac{k \cdot \log^2 n}{\log k}\right)$ for all $n, k \geq 2$. This unifies and improves upon two prior results: a lower bound of $\Omega(\log^2 n)$ from [13] and a lower bound of $\Omega\left(\frac{k \cdot \log n}{\log k}\right)$ from [4], respectively.

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

Keywords and phrases Tarski's theorem, monotone functions, lattices, fixed points, computational complexity, oracle model, query complexity, lower bounds

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2025.52

Category RANDOM

Related Version *Full Version*: `https://arxiv.org/abs/2502.16679` [5]

Funding This research was supported by the US National Science Foundation CAREER grant CCF-2238372.

1 Introduction

Let $\mathcal{L}$ be a complete lattice under the $\leq$ relation. A function $f : \mathcal{L} \to \mathcal{L}$ is monotone if $x \leq y$ implies $f(x) \leq f(y)$ for all $x, y \in \mathcal{L}$. Tarski's theorem [21] states that every such function has a fixed point. Tarski's theorem has applications in many areas. In game theory, pure Nash equilibria in supermodular games can be viewed as fixed points of monotone functions, and hence by Tarski's theorem such equilibria exist [13]. In denotational semantics, the semantics of recursively defined programs can be characterized as the least fixed points of monotone functions, ensuring a well-defined interpretation of recursion [16].

Recently, the computational complexity of finding Tarski fixed points has been investigated for grids. Specifically, for $k, n \in \mathbb{N}$, consider the k-dimensional grid $\{0, \ldots, n-1\}^k$ of side length n. Let $\leq$ be the binary relation where for each pair of vertices $a = (a_1, \ldots, a_k) \in \{0, \ldots, n-1\}^k$ and $b = (b_1, \ldots, b_k) \in \{0, \ldots, n-1\}^k$, we have $a \leq b$ if and only if $a_i \leq b_i$ for each $i \in [k]$.

* Authors are in alphabetical order.

In the query model, there is an unknown monotone function $f : \{0, \ldots, n-1\}^k \to \{0, \ldots, n-1\}^k$. The problem $TARSKI(n, k)$ is to find a fixed point of f given query access to the function, where the answer to each query $v \in \{0, \ldots, n-1\}^k$ is $f(v)$. The randomized query complexity is the minimum expected number of queries required to find a solution with a probability at least 9/10, where the expectation is taken over the coin tosses of the algorithm.

There are two main algorithmic approaches for finding a Tarski fixed point on grids. One approach is a path-following method that uses $O(nk)$ queries, which is best for small n and large k. For large n and small k, a divide-and-conquer approach works much better. Originally proposed by [10] and subsequently improved by [15] and [7], it achieves an upper bound of $O\left((\log n)^{\lceil \frac{k+1}{2} \rceil}\right)$ queries for any constant k.

There are a few known lower bounds which are optimized for different parameter regimes. [13] proved a randomized query complexity lower bound of $\Omega\left(\log^2(n)\right)$ for $TARSKI(n, 2)$. The same lower bound applies to $TARSKI(n, k)$ for $k \geq 2$. This lower bound matches the known upper bound for $k = 2$ and $k = 3$, though it does not scale with k. [4] proved two dimension-dependent randomized lower bounds, of $\Omega(k)$ and $\Omega\left(\frac{k \log n}{\log k}\right)$, respectively, for $TARSKI(n, k)$. These bounds scale with k, so they are stronger than [13] when k grows at about polylog(n). In particular, they characterize the randomized query complexity of finding a Tarski fixed point on the Boolean hypercube $\{0, 1\}^k$ (i.e. the power set lattice) as $\Theta(k)$. However, they are weaker than [13] for very small k.

In this paper, we show that the randomized query complexity of $TARSKI(n, k)$ is $\Omega\left(\frac{k \log^2(n)}{\log k}\right)$ for all $n, k \geq 2$. This is a dimension-dependent lower bound that unifies and improves upon the lower bounds of $\Omega\left(\log^2(n)\right)$ from [13] and of $\Omega\left(\frac{k \log n}{\log k}\right)$ from [4].

1.1 Our Contribution

Our main result is the following theorem.

▶ **Theorem 1.** *There exists $c > 0$ such that for all $k, n \in \mathbb{N}$ with $k, n \geq 2$, the randomized query complexity of $TARSKI(n, k)$ is at least $c \cdot \frac{k \log^2 n}{\log k}$.*

Proof Overview. Our first contribution is to design a multi-dimensional family $\mathcal{F}$ of herringbone functions, which generalizes the 2D construction from [13].

At a high level, there is a path ("spine") that runs from vertex $(0, \ldots, 0)$ to $(n-1, \ldots, n-1)$ that contains the unique fixed point. Along the spine, function values flow directly towards the fixed point. Outside the spine, the function is designed to be consistent with any position of the fixed point along the spine, which forces the algorithm to find "many" spine vertices to succeed. The challenge we overcome is defining such a function in a way that explicitly uses all the dimensions without violating monotonicity.

To obtain a lower bound, we invoke Yao's lemma, which allows focusing on a deterministic algorithm $\mathcal{A}$ that receives inputs drawn from the uniform distribution $\mathcal{U}$ over functions from $\mathcal{F}$. The high-level idea is that distribution $\mathcal{U}$ has the property that the location of each spine vertex is independent of spine vertices more than a short distance away. Consequently, a successful algorithm must repeatedly find spine vertices without relying on previously discovered ones, and finding each new spine vertex requires "many" queries.

Formalizing this intuition requires careful handling. We design a monotonic measure of progress that starts with a low value, increases by a small amount in expectation with each query, and must arrive at a high value for $\mathcal{A}$ to succeed. To do so, we first divide the space around the spine into regions. For each region R, we define random variables:

- $P_t(R) = \max_{v \in R} \log_2 \Pr[v \text{ is a spine vertex after } t \text{ queries from } \mathcal{A}]$ for all $t \in \mathbb{N}$.
- $P_t^*(R) = \max_{0 \le t^* \le t} P_{t^*}(R)$ for all $t \in \mathbb{N}$.

For $m \in \Theta(\log n)$, let G_m be the set of m-tuples of regions that are sufficiently far apart from each other that the location of the spine in each region is independent of the others in the tuple. Define

- $\overline{P}_t = \max_{(R^1,\ldots,R^m) \in G_m} \sum_{i=1}^{m} P_t^*(R^i)$ for all $t \in \mathbb{N}$.

The proof shows that $\overline{P}$ is a monotonic measure of progress with the required properties, which implies a lower bound on the number of queries needed for the algorithm to succeed.

Roadmap to the paper

The remainder of the paper is organized as follows. Related work is in Section 1.2. The model and preliminaries are in Section 2.

Our family $\mathcal{F}$ of multi-dimensional herringbone functions can be found in Section 3, with proofs of its properties in the full version ([5], Appendix A).

In Section 4 we define a distribution $\mathcal{U}$ on the family of functions $\mathcal{F}$, with proofs of its properties in the full version ([5], Appendix B). The proof of Theorem 1 can be found in Section 5, with proofs of supporting lemmas in the full version ([5], Appendix C). In Section 6 we also give upper bounds on the query complexity of finding a fixed point of any multi-dimensional herringbone function.

1.2 Related Work

Algorithms for the problem of finding Tarski fixed points on the k-dimensional grid of side length n have only recently been considered. [11] gave an $O(\log^k(n))$ divide-and-conquer algorithm. [15] gave an $O(\log^2(n))$ algorithm for the 3D grid and used it to construct an $O(\log^{2\lceil k/3 \rceil}(n))$ algorithm for the k-dimensional grid of side length n. [7] extended their ideas to get an $O(\log^{\lceil (k+1)/2 \rceil}(n))$ algorithm.

[13] showed a lower bound of $\Omega(\log^2(n))$ for the 2D grid, implying the same lower bound for the k-dimensional grid of side length n. This bound is tight for $k = 2$ and $k = 3$, but there is an exponential gap for larger k. They also showed that the problem is in both PLS and PPAD, which by the results of [14] implies it is in CLS.

[8] give a black-box reduction from the Tarski problem to the same problem with an additional promise that the input function has a unique fixed point. This result implies that the Tarski problem and the unique Tarski problem have the same query complexity.

Two problems related conceptually to that of finding a Tarski fixed point are finding a Brouwer fixed point [17, 6, 9] and finding a local minimum (i.e. local search) [2, 1, 23, 18, 20, 19, 12, 3]. The query complexity lower bounds for Brouwer and local search also rely on hidden path ("spine") constructions. However, the monotonicity condition of the function in the Tarski setting poses an extra challenge.

2 Model

Let $\{0, 1, \ldots, n-1\}^k$ be the k-dimensional grid of side length n under the $\le$ relation. Given oracle access to an (unknown) monotone function $f : \{0, 1, \ldots, n-1\}^k \to \{0, 1, \ldots, n-1\}^k$, the Tarski search problem, $TARSKI(n, k)$, is to find a fixed point of f using as few oracle queries as possible.

▶ **Definition 2** ($TARSKI(n,k)$). *Let $k, n \in \mathbb{N}$. Given oracle access to an unknown monotone function $f : \{0,1,\ldots,n-1\}^k \to \{0,1,\ldots,n-1\}^k$, find a vertex $x \in \{0,1,\ldots,n-1\}^k$ with $f(x) = x$ using as few queries as possible.*

Query complexity

The deterministic query complexity of a task is the total number of queries necessary and sufficient for a correct deterministic algorithm to find a solution. The randomized query complexity is the expected number of queries required to find a solution with probability at least 9/10 for each input[1], where the expectation is taken over the coin tosses of the algorithm.

Given an algorithm $\mathcal{A}$ for $TARSKI(n,k)$ that receives an input drawn from some input distribution $\mathcal{D}$, we say that an execution of $\mathcal{A}$ *succeeds* if it outputs a fixed point of the function given as input and *fails* otherwise.

Notation

The following notation is used throughout the paper.

- For a vector $\mathbf{x} \in \mathbb{R}^k$ and $i \in [k]$, we write the i-th coordinate of $\mathbf{x}$ as x_i.
- For a vector $\mathbf{x} \in \mathbb{R}^k$, we use $wt(\mathbf{x})$ to denote its Hamming weight $\sum_{i=1}^{k} x_i$.
- For each $j \in [k]$, let $\mathbf{e}_j = (0,\ldots,0,1,0,\ldots,0)$, where the vector $\mathbf{e}_j$ is all zeroes except the j-th coordinate in which it takes value 1.
- For all $u, v \in \mathbb{R}^k$, the line segment between them is $\ell(u,v) = \{\lambda u + (1-\lambda)v \mid \lambda \in [0,1]\}$.

3 Multi-dimensional Herringbone Functions

Our first contribution is to generalize the lower bound construction from [13] to any number of dimensions. Towards this end, we need to state the definition of a spine, which is a monotone sequence of vertices that each have Hamming distance 1 from their neighbors.

▶ **Definition 3** (Spine). *A spine $\mathbf{s} = \{s^0,\ldots,s^{(n-1)k}\}$ is a sequence of vertices with the property that $s^0 = (0,\ldots,0)$, $s^{(n-1)k} = (n-1,\ldots,n-1)$, and $s^{i+1} > s^i$ for all $i \in \{0,\ldots,(n-1)k-1\}$.*

▶ **Lemma 4.** *Let $\mathbf{s} = (s^0,\ldots,s^{(n-1)k})$ be a spine. Then $wt(s^i) = i \ \forall i \in \{0,\ldots,(n-1)k\}$.*

Proof. By the definition of a spine, we have $s^{i+1} > s^i$ for all $i \in \{0,\ldots,(n-1)k-1\}$. Thus $wt(s^{i+1}) > wt(s^i)$. The available Hamming weights are $\{0,\ldots,k(n-1)\}$, thus $wt(s^i) = i$ for all $i \in \{0,\ldots,(n-1)k\}$. ◀

▶ **Definition 5** (Multi-dimensional Herringbone Function). *Consider a spine $\mathbf{s} = \{s^0,\ldots,s^{(n-1)k}\}$ and an index $j \in \{0,\ldots,(n-1)k\}$. Define functions $M, \mu : \{0,\ldots,n-1\}^k \to \{0,\ldots,(n-1)k\}$ as follows:*

- $M(v) = \min\{i \in \{0,\ldots,(n-1)k\} \mid s^i \geq v\}$
- $\mu(v) = \max\{i \in \{0,\ldots,(n-1)k\} \mid s^i \leq v\}$.

[1] Any other constant greater than 1/2 would suffice.

Let $h^{\mathbf{s},j} : \{0,\dots,n-1\}^k \to \{0,\dots,n-1\}^k$ *be the function:*

$$h^{\mathbf{s},j}(v) = \begin{cases} v & \text{if } v = s^j \\ s^{i+1} & \text{if } v = s^i \text{ for some } i < j \\ s^{i-1} & \text{if } v = s^i \text{ for some } i > j \\ v + (s^{\mu(v)+1} - s^{\mu(v)}) - (s^{M(v)} - s^{M(v)-1}) & \text{otherwise.} \end{cases} \tag{1}$$

The first three cases of Definition 5 are the same as in the two-dimensional construction of [13], as they naturally extend to the higher-dimensional setting without modification.

Our key innovation lies in the fourth case, which applies to vertices not on the spine. It is designed to hide the location of s^j by being consistent with any value of j.

We show that each function $h^{\mathbf{s},j}$ from Definition 5 has a unique fixed point at s^j and is monotone. The proofs are included in the full version ([5], Appendix A).

▶ **Lemma 6.** *The function* $h^{\mathbf{s},j}$ *in Definition 5 has a unique fixed point at* s^j*.*

▶ **Lemma 7.** *The function* $h^{\mathbf{s},j}$ *in Definition 5 is monotone.*

On the spine, $h^{\mathbf{s},j}$ follows the spine to s^j. Off the spine, $h^{\mathbf{s},j}(v)$ increases in the dimension that the spine moves after $s^{\mu(v)}$ and decreases in the dimension that the spine moves before $s^{M(v)}$.

In two dimensions, this coincides with the herringbone function of [13]: vertices above the spine go down-right, and vertices below the spine go up-left. In higher dimensions, however, the off-spine vertices do not in general point directly at the spine.

4 Hard Distribution

In this section we define a hard distribution of inputs that is used to prove Theorem 1. We will use multi-dimensional herringbone functions with spines of a particular form. The proofs of statements in this section can be found in the full version ([5], Appendix B).

The idea is to restrict the spine to a "tube" running from the minimum to the maximum vertex in the lattice with width poly(n). We select connecting points on several slices throughout the tube and make the spine interpolate between these points.

We first introduce the tube and some of its useful properties.

▶ **Definition 8** (Tube)**.** *The tube of width* L *in* $\{0,\dots,n-1\}^k$ *is the set of points:*

$$T_L = \{x \in \{0,\dots,n-1\}^k \mid \exists i \in \{0,\dots,n-1\} \text{ such that } |x_j - i| \le L \ \forall j \in [k]\}. \tag{2}$$

The main appeal of the tube is that queries inside the tube can only provide information about nearby spine vertices.

▶ **Lemma 9.** *For a point* $v \in T_L$ *and a multi-dimensional herringbone function* f *whose spine lies in* T_L*, the value of* $f(v)$ *depends only on spine vertices whose coordinates are all within* $2L$ *of* v*'s.*

However, an algorithm is free to query any vertices, not just ones in the tube. We get around this by reducing a query to a vertex v outside the tube to two queries on the boundary of the tube, which together provide the same information as querying v.

▶ **Lemma 10.** *Let $L \in \mathbb{N}$ and $f : \{0, \ldots, n-1\}^k \to \{0, \ldots, n-1\}^k$ be a multi-dimensional herringbone function whose spine lies entirely in T_L. For each point $v \notin T_L$, there exist $a, b \in T_L$ such that:*

- *a and b can be computed from v and L; and*
- *$f(v)$ can be computed from $f(a)$ and $f(b)$.*

We now divide the tube T_L into regions, whose sizes are defined using a parameter $\rho \geq 0$ such that $\rho \mid k(n-1)$, $k \mid \rho$, and $\rho \geq 12kL$. We will eventually use $L = \frac{1}{12}\sqrt{n-1}$ and $\rho = k\sqrt{n-1}$, but these precise values are not important until the final bound is evaluated. We do require L and ρ to be integers, which will only occur if $n-1$ is the square of an integer that is divisible by 12. However, such values of n are sufficiently common that rounding down to the nearest one is a negligible loss.

We define a set of indices a for the regions we focus on as follows:

$$\mathcal{I} = \left\{ a \mid a \in \{0, \ldots, k(n-1)\} \text{ and } \rho \mid a \right\}. \tag{3}$$

▶ **Definition 11** (Regions). *For each $a \in \mathcal{I} \setminus \{k(n-1)\}$, define the region*

$$R_a = \{\mathbf{x} \in T_L \mid a \leq wt(\mathbf{x}) \leq a + \rho\}. \tag{4}$$

For all $a \in \mathcal{I}$, let $Low_a = \{\mathbf{x} \in T_L \mid wt(\mathbf{x}) = a\}$. For each region R_a, Low_a is its lower boundary.

The spines of our hard input distributions will be defined by passing through a series of connecting points, one at each region boundary.

▶ **Definition 12** (Connecting points). *Given $i \in \mathcal{I}$, the vertices χ_i are called* connecting points *if $\chi_i \in Low_i$.*

Given a sequence of connecting points χ_i, we next construct a spine that interpolates between them. Each χ_i represents the vertex that the spine passes through when entering region R_i from $R_{i-\rho}$.

For each vertex $v \in \{0, \ldots, n-1\}^k$, let $\mathcal{C}(v)$ be the axis-aligned cube of side length 1 centered at v, including its boundary faces. For each $i \in [k]$, the face of $\mathcal{C}(v)$ with minimum value in coordinate i is called *backward* and the face with maximum value in coordinate i is called *forward*.

▶ **Lemma 13.** *Let $a, b \in \{0, \ldots, n-1\}^k$ with $a \leq b$. For an arbitrary vertex $v \in \{0, \ldots, n-1\}^k$, suppose the cube $\mathcal{C}(v)$ intersects the line segment $\ell(a, b)$. Let $z \in \mathbb{R}^k$ be such that $z = \max\{\mathcal{C}(v) \cap \ell(a, b)\}$. Then z is well-defined, and either $v = b$ or z lies on some forward face of $\mathcal{C}(v)$.*

Using Lemma 13 as a subroutine, we can construct a path that follows the continuous line segment $\ell(u, v)$ between two given vertices u and v as closely as possible. In particular, each vertex s^i on the path will include some of the points in $\ell(u, v)$ in its cube $\mathcal{C}(s^i)$.

▶ **Lemma 14.** *Let $u, v \in \{0, \ldots, n-1\}^k$ be vertices with $u \leq v$. Then there exists a sequence of vertices $s^1, \ldots, s^m$ for some $m \in \mathbb{N}$ such that*

1. *$s^1 = u$ and $s^m = v$.*
2. *$(s^1, \ldots, s^m)$ is a monotone connected path in the graph $\{0, \ldots, n-1\}^k$.*
3. *For each $i \in [m]$, the set $\mathcal{C}(s^i) \cap \ell(u, v)$ is nonempty.*

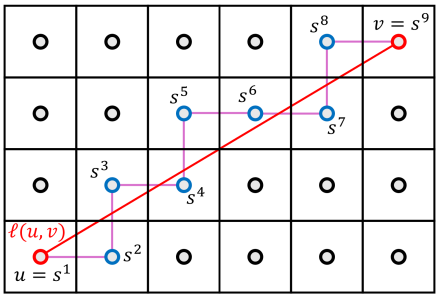

Figure 1 A 2-dimensional example of a sequence given by Lemma 14, with endpoints u and v and the associated spine vertices s^1 through s^9. Each vertex is drawn surrounded by its cube.

An illustration of a sequence given by Lemma 14 can be found in Figure 1.

A sequence of vertices $s = (s^1, \ldots, s^q)$ is *lexicographically smallest* among a set $\mathcal{W}$ of sequences if s has the smallest (lexicographically) possible first vertex s^1 among all choices for s^1 in $\mathcal{W}$, and among sequences tied in the first vertex, the lexicographically smallest possible second vertex s^2, and so forth.

▶ **Definition 15** (Spine induced by connecting points). *Let χ_i for $i \in \mathcal{I}$ be connecting points. For each consecutive pair of connecting points χ_i and $\chi_{i+\rho}$, define the sequence $(s^i, s^{i+1}, \ldots, s^{i+\rho})$ as the lexicographically smallest one that satisfies the properties of Lemma 14 invoked with $u = \chi_i$ and $v = \chi_{i+\rho}$. Let the spine be $\mathbf{s} = (s^0, \ldots, s^{(n-1)k})$.*

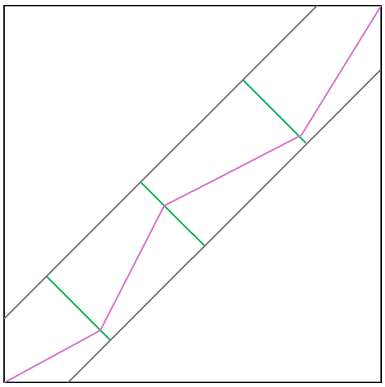

Figure 2 A sketch of the structure of the spines in Definition 15. The gray lines represent the edges of the tube T_L. The green lines represent Low_a for various $a \in \mathcal{I}$, and divide T_L into several regions. The pink path represents a possible spine through a choice of connecting points.

▶ **Observation 16.** *Each spine $\mathbf{s}$ from Definition 15 is monotone.*

Proof. Given a set of connecting points χ_i for $i \in \mathcal{I}$, the spine from Definition 15 is obtained by generating a monotone path P_i to connect each consecutive pair of connecting points χ_i and $\chi_{i+\rho}$. Then the spine is the union of the P_i's and thus monotone as well. ◀

▶ **Definition 17** (Set of spines Ψ, family of functions $\mathcal{F}$, and distribution $\mathcal{U}$). *The set of spines we consider is*

$$\Psi = \left\{ \mathbf{s} \mid \exists \text{ connecting points } \chi_i \text{ for } i \in \mathcal{I} \text{ such that } \mathbf{s} \text{ is the spine induced by them} \right\} \quad (5)$$

The family of functions we consider is $\mathcal{F} = \left\{ h^{\mathbf{s},j} \mid \mathbf{s} \in \Psi, j \in \{0, \ldots, k(n-1)\} \right\}$. Let $\mathcal{U}$ be the uniform distribution over functions in $\mathcal{F}$.

Choosing spine vertices independently in different regions allows us to show that queries within the tube truly give only local information, extending Lemma 9.

▶ **Lemma 18.** *Suppose a vertex $v \in \{0, \dots, n-1\}^k$ is in a region R_α. On input $f \sim \mathcal{U}$, the value of $f(v)$ is independent of the location of spine vertices in all regions except possibly regions $R_{\alpha+i\rho}$ for $i \in \{-2, \dots, 2\}$.*

5 Proof of the Main Theorem

We now move towards the proof of Theorem 1. At a high level, the lower bound in this theorem comes from a combination of two ideas:

- Lemma 24, which states that finding spine vertices in many different regions is difficult. Its proof is based on spine vertices being exponentially rare due to our construction.
- Lemma 25, which states that any correct algorithm for $TARSKI(n, k)$ must nevertheless find spine vertices in many different regions. Its proof is based on a reduction to ordered search.

The proofs of statements in this section are in the full version ([5], Appendix C).

We begin with the ideas leading to Lemma 24. The general idea is that any given vertex is very unlikely to be a spine vertex (Lemmas 19 and 20), while each query only provides $O(\log k)$ bits of information about the spine (Lemma 21).

We show that the number of points on each boundary between regions is exponentially large.

▶ **Lemma 19.** *Let $a \in \mathcal{I} \setminus \{0, k(n-1)\}$. Then $|Low_a| \geq (2L+1)^{\lfloor k/2 \rfloor}$.*

Because the number of points on each boundary is large, randomly selecting connecting points on the two ends of region doesn't concentrate the spine anywhere in that region.

▶ **Lemma 20.** *Let $w \in T_L$ be a vertex in the tube and $a, b \in \mathcal{I} \setminus \{0, k(n-1)\}$ such that $b - a \geq 12kL$. For random vertices $U \sim \mathcal{U}(Low_a)$ and $V \sim \mathcal{U}(Low_b)$, we have*

$$\Pr\Big[w \in \mathbf{s}(U, V)\Big] \leq \frac{17^k}{(2L+1)^{\lfloor k/2 \rfloor}}, \tag{6}$$

where $\mathbf{s}(U, V)$ is the lexicographically smallest connected monotone path from U to V that satisfies the properties of Lemma 14.

The next lemma shows that for each vertex v, the number of possible responses to querying v is $poly(k)$. This follows from Definition 5 and shows that each query only provides $O(\log k)$ bits of information.

▶ **Lemma 21.** *For each vertex $v \in \{0, \dots, n-1\}^k$, let*

$$F(v) = \left\{ y \in \{0, \dots, n-1\}^k \mid \exists \mathbf{s} \in \Psi, j \in \{0, \dots, k(n-1)\} \text{ such that } h^{\mathbf{s},j}(v) = y \right\}. \tag{7}$$

Then, for $k \geq 2$, we have $|F(v)| \leq k^3$.

Our central argument will revolve around finding spine vertices in "far-apart" regions. To that end, we define a distance between regions.

▶ **Definition 22.** *Given two regions R_α and R_β, their distance is defined as: $dist(R_\alpha, R_\beta) = |\alpha - \beta|/\rho$.*

Next we define what it means for an algorithm to "survey the spine".

▶ **Definition 23** (Surveying the spine). *On an input function $f \in \mathcal{F}$, an algorithm $\mathcal{A}$ is said to* m-survey the spine of f *if the following condition is met:*

- *the set of vertices queried by $\mathcal{A}$ contains m spine vertices $v_1, \ldots, v_m$ such that $dist(R^i, R^j) \geq 5$ for all $i \neq j$, where R^a is the region containing v_a.*

Now we can bound the success probability of algorithms that manage to survey the spine within a "few" queries.

▶ **Lemma 24.** *Let $m \in \mathbb{N}$. Let $\mathcal{A}$ be a deterministic algorithm that has oracle access to an unknown function $f \in \mathcal{F}$, where $\mathcal{A}$ is said to* succeed *on input f if it m-surveys the spine of f. Then if $\mathcal{A}$ makes at most T queries, its success probability on inputs drawn from $\mathcal{U}$ is upper bounded by:*

$$\Pr_{f \sim \mathcal{U}}[\mathcal{A} \text{ succeeds in } T \text{ queries on input } f] \leq \frac{15T \log_2(k)}{(m-2) \cdot (\lfloor k/2 \rfloor \log_2(2L+1) - k \log_2 17)} . \tag{8}$$

Proof sketch. We provide a proof sketch here; the complete proof can be found in the full version of the paper ([5], Appendix C).

The main idea is to find a measure of progress and show that the algorithm cannot progress too quickly. Let T be the number of queries issued by algorithm $\mathcal{A}$. For each time $t \in \{0, \ldots, T\}$, we define a random variable $\overline{P}_t$ such that:

- $\overline{P}_0$ is "small"
- the expectation of $\overline{P}_{t+1} - \overline{P}_t$ is bounded from above, and
- the final value $\overline{P}_T$ must be "large" for $\mathcal{A}$ to have a high success probability.

These properties collectively show a lower bound on the number of time steps T required for algorithm $\mathcal{A}$ to succeed with high probability.

To make this intuition precise, we define for each region R the following random variables:

- $P_t(R) = \max_{v \in R} \log_2 \Pr[v \text{ is a spine vertex after } t \text{ queries from } \mathcal{A}]$.
- $P_t^*(R) = \max_{0 \leq t^* \leq t} P_{t^*}(R)$.

Let $G_m = \{(R^1, \ldots, R^m) \mid R^1, \ldots, R^m \text{ are regions, } dist(R^i, R^j) \geq 5 \text{ for all } i \neq j \in [m]\}$.

That is, G_m is the set of m-tuples of regions which $\mathcal{A}$ is trying to find spine vertices in. Finally, define

- $\overline{P}_t = \max_{(R^1, \ldots, R^m) \in G_m} \sum_{i=1}^m P_t^*(R^i)$.

The proof shows that $\mathbb{E}[\overline{P}_{t+1} - \overline{P}_t] \leq 15 \log_2(k)$ for all t. This implies that if $\mathcal{A}$ makes at most T queries, then $\mathbb{E}[\overline{P}_T - \overline{P}_0] \leq 15T \log_2(k)$.

When $\mathcal{A}$ succeeds, it has found m spine vertices in regions at least five apart, and therefore $\overline{P}_T = 0$. By Lemma 20, no vertex (other than those in the first and last regions, which we can effectively ignore) initially has a chance higher than $17^k/(2L+1)^{\lfloor k/2 \rfloor}$ to be on the spine, which upper-bounds $\overline{P}_0$. Applying Markov's inequality to $\overline{P}_T - \overline{P}_0$ then upper-bounds $\mathcal{A}$'s success probability. ◀

▶ **Lemma 25.** *Let $\mathcal{A}$ be a deterministic algorithm for $TARSKI(n, k)$ that receives an input drawn from $\mathcal{U}$. Let $\delta, \epsilon \in (0, 1)$ be such that $2\delta + 2\epsilon < 1$. Suppose $k(n-1)/\rho > 2^{3/(1-2\delta-2\epsilon)}$.*

There exists $c = c(\delta, \epsilon)$ such that if $\Pr[\mathcal{A} \text{ succeeds}] \geq 1 - \delta$, then $\mathcal{A}$ queries spine vertices in at least $c \log(k(n-1)/\rho)$ regions with probability at least ϵ.

Proof sketch. We provide a proof sketch here; the complete proof can be found in the full version ([5], Appendix C).

By the construction of a multi-dimensional herringbone function, finding the index j of the fixed point s^j appears to require solving ordered search on the other spine vertices. We formalize this by constructing a randomized algorithm $\mathcal{A}'$ for ordered search that works by simulating $\mathcal{A}$ on an input with a synthetic, random spine. When $\mathcal{A}'$ is itself run on a uniform distribution over ordered search instances, the simulated $\mathcal{A}$ is run on inputs from $\mathcal{U}$. We then use lower bounds on ordered search to complete the proof. ◀

We can now prove a lower bound on the randomized query complexity of $TARSKI(n, k)$.

Proof of Theorem 1. We proceed by invoking Yao's lemma. Let $\mathcal{U}$ be the uniform distribution over the set of functions $\mathcal{F}$. Let $\mathcal{A}$ be the deterministic algorithm with the smallest possible expected number of queries that succeeds with probability at least $4/5$, where both the expected query count and the success probability are for input drawn from $\mathcal{U}$. The algorithm $\mathcal{A}$ exists since there is a finite number of deterministic algorithms for this problem[2].

Let D be the expected number of queries issued by $\mathcal{A}$ on input drawn from $\mathcal{U}$. Let R be the randomized query complexity of $TARSKI(n, k)$; i.e. the expected number of queries required to succeed with probability at least $9/10$. Then Yao's lemma ([22], Theorem 3) yields $2R \geq D$. Therefore it suffices to lower bound D.

We argue next that up to a factor of 2, we can assume that all queries are inside the tube T_L. Let $\overline{TARSKI}(n, k)$ be the Tarski search problem where the input is drawn uniformly at random from $\mathcal{F}$ and the algorithm is only allowed to query vertices from the tube. By Lemma 10, whenever $\mathcal{A}$ queries a vertex v outside the tube, it could find instead two vertices a, b inside the tube such that the answers to queries a and b carry at least as much information as the answer to query v. Therefore, we can transform $\mathcal{A}$ into an algorithm $\overline{\mathcal{A}}$ for $\overline{TARSKI}(n, k)$ that makes at most $2D$ queries in expectation.

Thus from now on we lower bound the expected number of queries of algorithms that are only allowed to query vertices in the tube.

The success probability of $\overline{\mathcal{A}}$ on inputs drawn from $\mathcal{U}$ is at least $4/5$. By Lemma 25, for $\epsilon = 1/5$, there exists $c = c(\epsilon)$ such that, with probability at least ϵ, algorithm $\overline{\mathcal{A}}$ queries spine vertices in at least $c \log(k(n-1)/\rho)$ regions. Whenever it does so, taking every fifth region that $\overline{\mathcal{A}}$ queries a spine vertex in gives at least $m = \left\lfloor \frac{c}{5} \log(k(n-1)/\rho) \right\rfloor$ regions, all of which are distance at least 5 from each other; in other words, algorithm $\overline{\mathcal{A}}$ has m-surveyed the spine. Therefore:

$$\Pr_{f \sim \mathcal{U}} \left[\overline{\mathcal{A}} \text{ } m\text{-surveys the spine of } f \right] \geq \epsilon \,. \tag{9}$$

By Lemma 24 applied to algorithm $\overline{\mathcal{A}}$, for any $T \in \mathbb{N}$:

$$\begin{aligned} &\Pr_{f \sim \mathcal{U}} \left[\overline{\mathcal{A}} \text{ } m\text{-surveys the spine within } T \text{ queries on input } f \right] \\ &\quad \leq \frac{15T \log_2(k)}{(m-2)\left(\lfloor k/2 \rfloor \log_2(2L+1) - k \log_2 17 \right)} \,. \end{aligned} \tag{10}$$

[2] Strictly speaking, there are infinitely many algorithms if we allow querying the same vertex multiple times. However, these algorithms are strictly worse than equivalent versions that query each vertex at most once, and there are only finitely many of those.

Applying (10) with

$$T_\epsilon = \frac{1}{30\log_2(k)} \cdot \epsilon\,(m-2) \cdot (\lfloor k/2 \rfloor \log_2(2L+1) - k\log_2 17)$$

gives:

$$\Pr_{f\sim\mathcal{U}}\left[\overline{\mathcal{A}}\ m\text{-surveys the spine within } T_\epsilon \text{ queries on input } f\right] \leq \frac{\epsilon}{2}\,. \tag{11}$$

By (9), we must therefore have

$$\Pr_{f\sim\mathcal{U}}\left[\overline{\mathcal{A}} \text{ makes more than } T_\epsilon \text{ queries on input } f\right] \geq \epsilon/2\,.$$

Thus the expected number D of queries $\overline{\mathcal{A}}$ makes on inputs drawn from $\mathcal{U}$ must be at least:

$$D \geq T_\epsilon \cdot \frac{\epsilon}{2} = \frac{\epsilon^2\left(\left\lfloor \frac{c}{5}\log(k(n-1)/\rho)\right\rfloor - 2\right)\cdot(\lfloor k/2\rfloor \log_2(2L+1) - k\log_2 17)}{60\log_2(k)}\,. \tag{12}$$

Since ϵ and c are constants, setting $\rho = k\sqrt{n-1}$ and $L = \frac{1}{12}\sqrt{n-1}$ in Equation (12) implies that $D \in \Omega(k\log^2(n)/\log(k))$, as desired. ◀

6 An $O(k\log n\log(nk))$ upper bound for multi-dimensional herringbone functions

We also provide a deterministic upper bound on the query complexity of finding a fixed point of any multi-dimensional herringbone function, which is quite close to the lower bound of Theorem 1. We include the theorem statement and proof sketch here; see the full version ([5], Appendix D) for the proof.

▶ **Theorem 26.** *There exists an $O(k\log(n)\log(nk))$-query algorithm to find the fixed point of a multi-dimensional herringbone function.*

Proof sketch. Our algorithm uses an $O(k\log n)$-query subroutine that finds, for an input x, the spine vertex with Hamming weight x. This subroutine exploits the dependence of the off-spine function values on the shape of the spine to perform k independent binary searches along the k axes. Using this subroutine, we can run binary search along the spine to find the fixed point, which takes $O(\log(nk))$ iterations. ◀

References

1. Scott Aaronson. Lower bounds for local search by quantum arguments. *SIAM J. Comput.*, 35(4):804–824, 2006. doi:10.1137/S0097539704447237.
2. David Aldous. Minimization algorithms and random walk on the d-cube. *The Annals of Probability*, 11(2):403–413, 1983.
3. Simina Brânzei, Davin Choo, and Nicholas J. Recker. The sharp power law of local search on expanders. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1792–1809. SIAM, 2024. doi:10.1137/1.9781611977912.71.
4. Simina Brânzei, Reed Phillips, and Nicholas J. Recker. The randomized query complexity of finding a tarski fixed point on the boolean hypercube. *CoRR*, abs/2409.03751, 2024. doi:10.48550/arXiv.2409.03751.
5. Simina Brânzei, Reed Phillips, and Nicholas Recker. Tarski lower bounds from multi-dimensional herringbones, 2025. doi:10.48550/arXiv.2502.16679.

6 Xi Chen and Xiaotie Deng. On algorithms for discrete and approximate brouwer fixed points. In *Proceedings of the thirty-seventh annual ACM symposium on Theory of computing*, pages 323–330, 2005. doi:10.1145/1060590.1060638.

7 Xi Chen and Yuhao Li. Improved upper bounds for finding tarski fixed points. In *Proceedings of the 23rd ACM Conference on Economics and Computation*, pages 1108–1118, 2022. doi:10.1145/3490486.3538297.

8 Xi Chen, Yuhao Li, and Mihalis Yannakakis. Reducing tarski to unique tarski (in the black-box model). In Amnon Ta-Shma, editor, *Computational Complexity Conference (CCC)*, volume 264, pages 21:1–21:23, 2023. doi:10.4230/LIPICS.CCC.2023.21.

9 Xi Chen and Shang-Hua Teng. Paths beyond local search: A tight bound for randomized fixed-point computation. In *48th Annual IEEE Symposium on Foundations of Computer Science (FOCS'07)*, pages 124–134. IEEE, 2007. doi:10.1109/FOCS.2007.14.

10 Chuangyin Dang, Qi Qi, and Yinyu Ye. Computational models and complexities of tarski's fixed points. Technical report, Stanford University, 2011.

11 Chuangyin Dang, Qi Qi, and Yinyu Ye. Computations and complexities of tarski's fixed points and supermodular games, 2020. arXiv:2005.09836.

12 Hang Dinh and Alexander Russell. Quantum and randomized lower bounds for local search on vertex-transitive graphs. *Quantum Information & Computation*, 10(7):636–652, 2010. doi:10.26421/QIC10.7-8-5.

13 Kousha Etessami, Christos Papadimitriou, Aviad Rubinstein, and Mihalis Yannakakis. Tarski's theorem, supermodular games, and the complexity of equilibria. In *Innovations in Theoretical Computer Science Conference (ITCS)*, volume 151, pages 18:1–18:19, 2020. doi:10.4230/LIPICS.ITCS.2020.18.

14 John Fearnley, Paul Goldberg, Alexandros Hollender, and Rahul Savani. The complexity of gradient descent: CLS = PPAD ∩ PLS. *Journal of the ACM*, 70:1–74, 2022. doi:10.1145/3568163.

15 John Fearnley, Dömötör Pálvölgyi, and Rahul Savani. A faster algorithm for finding tarski fixed points. *ACM Transactions on Algorithms (TALG)*, 18(3):1–23, 2022. doi:10.1145/3524044.

16 Stephen Forrest. The knaster-tarski fixed point theorem for complete partial orders, 2005. Lecture notes, McMaster University: http://www.cas.mcmaster.ca/~forressa/academic/701-talk.pdf.

17 Michael D Hirsch, Christos H Papadimitriou, and Stephen A Vavasis. Exponential lower bounds for finding brouwer fix points. *Journal of Complexity*, 5(4):379–416, 1989. doi:10.1016/0885-064X(89)90017-4.

18 Donna Crystal Llewellyn, Craig Tovey, and Michael Trick. Local optimization on graphs. *Discrete Applied Mathematics*, 23(2):157–178, 1989. doi:10.1016/0166-218X(89)90025-5.

19 Miklos Santha and Mario Szegedy. Quantum and classical query complexities of local search are polynomially related. In *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*, pages 494–501, 2004. doi:10.1145/1007352.1007427.

20 Xiaoming Sun and Andrew Chi-Chih Yao. On the quantum query complexity of local search in two and three dimensions. *Algorithmica*, 55(3):576–600, 2009. doi:10.1007/S00453-008-9170-6.

21 Alfred Tarski. A lattice-theoretical fixpoint theorem and its applications. *Pacific J. Math.*, 5:285–309, 1955.

22 Andrew Chi-Chih Yao. Probabilistic computations: Toward a unified measure of complexity. In *18th Annual Symposium on Foundations of Computer Science (sfcs 1977)*, pages 222–227, 1977. doi:10.1109/SFCS.1977.24.

23 Shengyu Zhang. Tight bounds for randomized and quantum local search. *SIAM Journal on Computing*, 39(3):948–977, 2009. doi:10.1137/06066775X.