48th Annual IEEE Symposium on Foundations of Computer Science

Paths Beyond Local Search: A Tight Bound for Randomized Fixed-Point Computation*

Xi Chen†
Department of Computer Science
Tsinghua University

Shang-Hua Teng‡
Department of Computer Science
Boston University

Abstract

In 1983, Aldous proved that randomization can speedup local search. For example, it reduces the query complexity of local search over grid $[1 : n]^d$ *from* $\Theta(n^{d-1})$ *to* $O(d^{1/2}n^{d/2})$*. It remains open whether randomization helps fixed-point computation. Inspired by the recent advances on the complexity of equilibrium computation, we solve this open problem by giving an asymptotically tight bound of* $(\Omega(n))^{d-1}$ *on the randomized query complexity for computing a fixed point of a discrete Brouwer function over grid* $[1 : n]^d$*. Our result can be extended to the black-box query model for Sperner's Lemma in any dimension. It also yields a tight bound for the computation of* d*-dimensional approximate Brouwer fixed points as defined by Scarf and by Hirsch, Papadimitriou, and Vavasis.*

Since the randomized query complexity of global optimization over $[1 : n]^d$ *is* $\Theta(n^d)$*, the randomized query model over* $[1 : n]^d$ *strictly separates these three important search problems:*

Global optimization is harder than fixed-point computation, and fixed-point computation is harder than local search.

Our result indeed demonstrates that randomization does not help much in fixed-point computation in the black-box query model. Our randomized lower bound matches the deterministic complexity of this problem, which is $\Theta(n^{d-1})$*.*

*This research is supported mostly by the NSF ITR grant CCR-0325630.

†Part of this work was done while visiting Computer Science Department at Boston University. In part supported by the National Natural Science Foundation of China Grant 60553001, the National Basic Research Program of China Grant (2007CB807900, 2007CB807901), and the Chinese National Key Foundation R & D Plan (2003CB317807, 2004CB318108).

‡Part of this work was done while visiting Tsinghua University and Microsoft Research Asia Lab.

1 Introduction

In this paper, we prove that randomization cannot be used to speedup fixed-point computation (FPC) in the black-box query model. Our result draws a striking contrast between FPC and local search: In 1983, Aldous proved that randomization can significantly speedup local search [2]. Our randomized lower bound is asymptotically tight — it matches the deterministic upper bound [18, 6] for this problem. Our result resolves a question that has been open since the 1989 paper of Hirsch, Papadimitriou, and Vavasis [18] which introduced the black-box query model for FPC.

Motivation

The Simplex Algorithm [13] is an implementation of local search[1] and finding a Nash equilibrium [25] is an example of FPC. A general approach for local search is Iterative Improvement with Steepest-Descent as its most popular example. It follows a path along which the objective values are improving, in the feasible space. The end of the path is a local optimum. Likewise, algorithms for FPC, such as the Lemke-Howson algorithm [24] and the algorithm for Sperner's Lemma [35], also follow a path whose endpoint is an equilibrium or a fixed-point. But unlike in local search, a path in FPC does not have an obvious locally computable, monotonic[2] measure-of-progress. Moreover, path following in FPC from an arbitrary point could lead to a cycle while the union of paths in Iterative Improvement is acyclic.

Do these structural differences have any algorithmic implication?

There has been increasing evidence that local search and FPC are different. First, Aldous [2] showed that randomization can speedup local search (details below). His method crucially utilizes the monotonicity discussed above.

[1] In linear programming, each local optimum is also a global optimum.

[2] Each path has a globally computable monotonic measure, the number of hops from the start of the path to a node.

0272-5428/07 $25.00 © 2007 IEEE
DOI 10.1109/FOCS.2007.14

IEEE computer society

Authorized licensed use limited to: Tsinghua University Library. Downloaded on November 24, 2008 at 00:45 from IEEE Xplore. Restrictions apply.

It remains open whether randomization helps FPC. Second, polynomial-time algorithms have been developed for several non-trivial classes of local search problems. These algorithms include the interior-point algorithm for linear and convex programming [22, 26] and edge-insertion algorithms for geometric optimization [16]. However, popular fixed-point problems such as the computation of Nash or market equilibria [3] might be hard for polynomial time [14, 8, 12]. Other than those that can be solved by convex programming, we haven't yet discovered a significantly non-trivial class of equilibrium problems solvable in polynomial time. Third, an approximate local optimum for each **PLS** (Polynomial Local Search [21]) problem can be found in fully-polynomial time [27]. In contrast, one can apply a fully-polynomial-time approximate Nash equilibrium algorithm to find an exact Nash equilibrium in polynomial time [9]. Same can be said about approximate market equilibria [19]. Fourth, although they all have exponential worst-case complexity [31, 23], the smoothed complexity of the Simplex Algorithm and Lemke-Howson Algorithm (or Scarf's market equilibrium algorithm [34]) might be drastically different [36, 9, 19]. This evidence inspires us to ask:

Is fixed-point computation fundamentally harder than local search?

Naturally, the best way to address this question is to prove "If **PPAD** is in **FP**, then **PLS** is in **FP**." This question remains open and there are oracles separating **PPAD** and **PLS** from each other [4]. Here, we show that in the black-box query model, FPC is strictly harder than local search.

Black-Box Query Model for FPC and Local Search

For local search, we will use the model studied by Aldous [2] and Aaronson [1] (see also [30, 39, 37]). The search space is defined over $\mathbb{Z}_n^d = [1:n]^d$ and we are given a black-box function $h : \mathbb{Z}_n^d \to \mathbb{R}$. The *local search problem* is to find a local optimum of h, a vector $\mathbf{x} \in \mathbb{Z}_n^d$ such that $h(\mathbf{x}) \geq h(\mathbf{y}), \forall\ \mathbf{y}$ with $\|\mathbf{x}-\mathbf{y}\|_1 \leq 1$. For FPC, we consider the model introduced by Hirsch, Papadimitriou and Vavasis [18]. In the HPV model, the search domain is also $\mathbb{Z}_n^d = [1:n]^d$. We are given a black-box function $F : \mathbb{Z}_n^d \to \mathbb{Z}_n^d$ that satisfies Brouwer's condition [5] — a set of continuity and boundary conditions (see Section 2) — that guarantees the existence of a fixed-point. The *FPC problem* is to find a fixed-point of F, a vector $\mathbf{v} \in \mathbb{Z}_n^d$ satisfying $F(\mathbf{v}) = \mathbf{v}$. The complexity of both problems is measured by the number of queries, of the form "What is $h(\mathbf{x})$?" or "What is $F(\mathbf{x})$?", needed to find a solution.

The HPV model is in fact a very good approximation for the computation of Brouwer's fixed points, as argued in [18]. To be self-contained, we include a brief discussion here. Note that Brouwer's Fixed Point Theorem [5] — that any continuous map f from a convex compact body, such as a simplex or a hypercube, to itself has a fixed point — is inherently continuous. In order to study complexity of this continuous problem in the Turing model, some inaccuracy must be introduced to ensure the existence of a solution with finite description [32, 33, 28, 18, 15].

Following Scarf [32], an approximate fixed point of a continuous map f is a point $\mathbf{x}$ in the convex body such that $\|f(\mathbf{x})-\mathbf{x}\| \leq \epsilon$ for a given $\epsilon > 0$. In 1928, Sperner [35] discovered a discrete fixed point theorem that led to the most elegant proof of Brouwer's Theorem. Suppose that Ω is a d-dimensional simplex with vertices $v_1, v_2, ..., v_{d+1}$, and that $\mathcal{S}$ is a simplicial decomposition of Ω. Suppose Π assigns to each vertex of $\mathcal{S}$ a color from $\{1, 2, ..., d+1\}$ such that, for every vertex v of $\mathcal{S}$, $\Pi(v) \neq i$ if the i^{th} component of the barycentric coordinate of v (the convex combination of $v_1, v_2, ..., v_{d+1}$ to express v), is 0. Sperner's Lemma asserts that there exists a cell in $\mathcal{S}$ that contains all colors. Consider a Brouwer map f with Lipschitz constant L over the simplex Ω. Suppose further that the diameter of each simplex cell in $\mathcal{S}$ is at most ϵ/L. Then, one can define a color assignment Π_f such that each fully-colored simplex in $(\mathcal{S}, \Pi_f)$ must have a vertex $\mathbf{v}$ satisfying $\|f(\mathbf{v}) - \mathbf{v}\| \leq \Theta(\epsilon)$. Thus, a fully-colored cell of $(\mathcal{S}, \Pi_f)$ can be viewed as an approximate, discrete fixed point of f. The HPV model is an extension of Sperner's Lemma from the simplex to the hypercube. Our randomized lower bound can be extended to the black-box query model for Sperner's Lemma and hence for the computation of approximate fixed points.

There are some similarities between FPC and local search over $\mathbb{Z}_n^d$. For both, divide-and-conquer has positive but limited success: It can solve both problems using $O(n^{d-1})$ queries [6]. An alternative approach to solve them is path-following. When following a short path, it can be faster than divide-and-conquer, but long and winding paths are the cause of inefficiency. The most prominent difference between paths in these two problems is that the values of h along a path to a local optimum are monotonic, serving as a measure-of-progress along the path. Aldous [2] used this fact in a randomized algorithm: Randomly query $d^{1/2}n^{d/2}$ points in $\mathbb{Z}_n^d$; let $\mathbf{s}$ be the sample point with the largest h value; follow a path starting at $\mathbf{s}$. If a path to a local optimum is long, say much longer than $d^{1/2}n^{d/2}$, then with high probability, the random samples intersect the path and partition it into sub-paths, each with expected length $O(d^{1/2}n^{d/2})$. As $\mathbf{s}$ has the largest h value, its sub-path is the last sub-path of a potentially long path, but with expected length $O(d^{1/2}n^{d/2})$. So with randomization, Aldous reduced the expected query complexity to $O(d^{1/2}n^{d/2})$. It remains open whether randomization can reduce the query complexity of FPC over $\mathbb{Z}_n^d$. The lack of a measure-of-progress along a path makes it impossible for us to directly use Aldous' idea.

Authorized licensed use limited to: Tsinghua University Library. Downloaded on November 24, 2008 at 00:45 from IEEE Xplore. Restrictions apply.

Our Main Result

As the main technical result of this paper, we prove that an expected number of $(\Omega(n))^{d-1}$ queries are indeed needed for FPC. Our lower bound is asymptotically tight[3] as a function of n, since the divide-and-conquer algorithm in [6] can find a fixed point by querying $O(n^{d-1})$ vectors.

In contrast to Aldous' result [2], our result demonstrates that randomization does not help much in FPC in the query model. It shows that, in the randomized query model over $\mathbb{Z}_n^d$, a fixed-point is strictly harder to find than a local optimum! The significant gap between these two problems is revealed only in randomized computation. In the deterministic framework, both have query complexity $\Theta(n^{d-1})$.

One can show that the randomized query complexity for finding a global optimum over $\mathbb{Z}_n^d$ is $\Theta(n^d)$. Thus, the randomized query model over $\mathbb{Z}_n^d$ strictly separates these three important search problems:

Global optimization is harder than fixed-point computation, and fixed-point computation is harder than local search.

Although the beautiful question "does **PPAD** in **FP** implies **PLS** is in **FP**?" remains open, our results uncover some fundamental difficulties in fixed-point and equilibrium computation. We also anticipate that a similar gap can be obtained in the quantum query model.

Our proof has two stages: We first define a string problem (Section 2.2), and reduce it to the FPC problem over $\mathbb{Z}_n^d$. We then prove a randomized lower bound for the string problem, which implies a same bound for FPC. The lower bound proof is obtained via a hierarchical construction of random long strings. Our reduction exposes the hardness of FPC, and allows us focus on the combinatorially simpler string problem. The idea behind the reduction is illustrated by Figure 1, in which we generate a simple graph G' over $[3:27]^2$ from a string "1537" of integers. Our string-based method systematically generates random, long and winding paths in the grid graph over $\mathbb{Z}_n^d$. To achieve our nearly-tight lower bound, these paths must be much longer than the random paths constructed in [39, 37] for local search. Our paths have expected length $(\Theta(n))^{d-1}$, while those paths for local search have length $\Theta(n^{d/2})$. We also develop new techniques for unknotting a self-intersecting path in $\mathbb{Z}_n^d$ and for realizing a path with a Brouwer function. These techniques are instrumental to our analysis and could be useful in the future complexity studies of FPC and its applications.

Related Work

Our work is partially inspired by the results of Aaronson [1], Santha and Szegedy [30], Zhang [39], and Sun and Yao [37] on the randomized and quantum query complexity of local search over $\mathbb{Z}_n^d$. It is built on the model of Hirsch, Papadimitriou and Vavasis [18], who proved a tight $\Theta(n)$ deterministic bound for $\mathbb{Z}_n^2$ and an $\Omega(n^{d-2})$ lower bound for $\mathbb{Z}_n^d$. Chen and Deng [6] improved this bound to $\Theta(n^{d-1})$ for $\mathbb{Z}_n^d$. Friedl, Ivanyos, Santha, and Verhoeven [17] gave an $\Omega(n^{1/4})$-lower bound on the randomized query complexity of the 2-dimensional Sperner problem. Our method for unknotting self-intersecting paths can be viewed as an extension of the 2D technique of [7] to high dimensions.

2 Three High-Dimensional Search Problems

We define three search problems. The first one concerns FPC. We introduce the last two to help the study of the first one. For each of the three problems, we define its mathematical structure, a query model for accessing this structure, the search problem itself, and its query complexity. Below, let $\mathbb{E}^d = \{\pm\mathbf{e}_1, \pm\mathbf{e}_2, ..., \pm\mathbf{e}_d\}$. Let $\|\cdot\|$ denote $\|\cdot\|_\infty$. For two vectors $\mathbf{u} \neq \mathbf{v}$ in $\mathbb{Z}^d$, we say $\mathbf{u} < \mathbf{v}$ lexicographically if $u_i < v_i$ for some i, and $u_j = v_j$ for all $1 \leq j < i$.

2.1 Discrete Brouwer Fixed-Points

A function $f : \mathbb{Z}_n^d \to \{\mathbf{0}\} \cup \mathbb{E}^d$ is *bounded* if $f(\mathbf{x}) + \mathbf{x} \in \mathbb{Z}_n^d$ for all $\mathbf{x} \in \mathbb{Z}_n^d$; $\mathbf{v} \in \mathbb{Z}_n^d$ is a *zero point* of f if $f(\mathbf{v}) = \mathbf{0}$. Clearly, if $F(\mathbf{x}) = \mathbf{x} + f(\mathbf{x})$, then $\mathbf{v}$ is a fixed point of F iff $\mathbf{v}$ is a zero point of f. A function $f : S \to \{\mathbf{0}\} \cup \mathbb{E}^d$, where $S \subset \mathbb{Z}^d$, is direction-preserving if $\|f(\mathbf{r}_1) - f(\mathbf{r}_2)\| \leq 1$ for all pairs $\mathbf{r}_1, \mathbf{r}_2 \in S$ such that $\|\mathbf{r}_1 - \mathbf{r}_2\| \leq 1$.

Following the discrete fixed-point theorem of [20], we have: For every function $f : \mathbb{Z}_n^d \to \{\mathbf{0}\} \cup \mathbb{E}^d$, if f is both bounded and direction-preserving, then there exists $\mathbf{v} \in \mathbb{Z}_n^d$ such that $f(\mathbf{v}) = \mathbf{0}$. We refer to a bounded and direction-preserving function f over $\mathbb{Z}_n^d$ as a *Brouwer function* over $\mathbb{Z}_n^d$. In the query model, one can only access f by asking queries of the form: "What is $f(\mathbf{r})$?" for a point $\mathbf{r} \in \mathbb{Z}_n^d$.

The FPC problem ZP^d that we will study is: *Given a Brouwer function $f : \mathbb{Z}_n^d \to \{\mathbf{0}\} \cup \mathbb{E}^d$ in the query model, find a zero point of f.* Given any randomized algorithm Υ[4] for solving ZP^d, we let $\mathsf{RQ}(\Upsilon, f)$ denote the expected number of queries used by Υ to output a zero point. We let

$$\mathsf{RQ}^d_{\mathsf{ZP}}(n) = \min_{\Upsilon} \max_{f:\text{ Brouwer function over } \mathbb{Z}_n^d} \mathsf{RQ}(\Upsilon, f)\ ,$$

denote the *randomized query complexity* of ZP^d. In this paper, we will prove:

Theorem 2.1 (Main)**.** *For all $d \geq 2$, there exists a constant c such that for all sufficiently large n,*

$$\mathsf{RQ}^d_{\mathsf{ZP}}(n) \geq (c^{-d}n)^{d-1}.$$

[3] The constant in Ω in our lower bound depends exponentially on d.

[4] We assume that Υ does not stop until a zero point of f is found.

Authorized licensed use limited to: Tsinghua University Library. Downloaded on November 24, 2008 at 00:45 from IEEE Xplore. Restrictions apply.

In contrast, the deterministic query complexity for solving ZP^d *is at most* $7n^{d-1}$ [6].

The FPC problem defined here is computationally equivalent to the fixed problems studied in [18, 14, 9, 35]. Thus, our result carries over to these FPC problems.

2.2 End-of-a-String

Suppose Σ is a finite set. A string S over Σ of length m is a sequence $S = a_1a_2...a_m$ with $a_i \in \Sigma$.

Definition 2.2 (Non-Repeating-Strings)**.** *A string* $S = a_1 a_2...a_m$ *over* $\mathbb{Z}_n = [1 : n]$ *is* d-non-repeating *for* $d \in [1 : m]$ *if* (1) *each string over* $\mathbb{Z}_n$ *of length* d *appears in* S *as a substring at most once;* (2) a_i *is odd if* i *is a multiple of* d *and* a_i *is even otherwise; and* (3) m *is a multiple of* d*.*

We define $\mathsf{end}_d(S) = a_{m-d+1}...a_{m-1}a_m$.

Each d-non-repeating string $S = a_1...a_m$ over $\mathbb{Z}_n$ defines a query oracle $\mathbb{B}_S : [1 : n]^d \to (\{\text{“no”}\} \cup \mathbb{Z}_n) \times (\{\text{“no”}\} \cup \mathbb{Z}_n)$: For $S' = b_1...b_d \in [1 : n]^d$, if S' is not a substring of S, then $\mathbb{B}_S(S') = (\text{“no”}, \text{“no”})$; otherwise, there is a unique k such that $a_{k+i-1} = b_i$, $\forall\ i \in [1 : d]$. Then $\mathbb{B}_S(S') = (\text{“no”}, a_{d+1})$ when $k = 1$, $\mathbb{B}_S(S') = (a_{m-d}, \text{“no”})$ when $k = m - d + 1$ (i.e., $S' = \mathsf{end}_d(S)$), and $\mathbb{B}_S(S') = (a_{k-1}, a_{k+d})$, otherwise.

Let ES^d denote the search problem: *Given a pair* (S, n) *where* S *is a* d*-non-repeating string over* $\mathbb{Z}_n$ *accessible by* $\mathbb{B}_S$[5]*, and its first* d *symbols* $a_1...a_{d-1}a_d$ *with* $a_d = 1$*, find* $\mathsf{end}_d(S)$. We let $\mathsf{RQ}^d_{\mathsf{ES}}(n)$ denote its randomized query complexity[6]. It is easy to show that $\mathsf{RQ}^1_{\mathsf{ES}}(n) = \Theta(n)$. In Section 4, we will prove

Theorem 2.3 (Complexity of ES^d)**.** *For all* $d \geq 2$ *and sufficiently large* n*,* $\mathsf{RQ}^d_{\mathsf{ES}}(4n + 4) \geq 2^{-(d+1)}\,(24^{-d}n)^d$.

In Section 3, we will reduce ES^{d-1} to ZP^d and prove Theorem 2.1 as a corollary of Theorem 2.3. In the reduction, the two problems are connected via the following problem on graphs.

2.3 End-of-a-Path in Grid-PPAD Graphs

The mathematical structure for this search problem is a directed graph $G = (V, E)$. A vertex $v \in V$ satisfies *Euler's condition* if $\Delta_I(v) = \Delta_O(v)$ where $\Delta_I(v)$ and $\Delta_O(v)$ are the in-degree and out-degree of v, respectively. We start with the following definition motivated by Papadimitriou's **PPAD** class [29].

Definition 2.4 (Generalized **PPAD** Graphs)**.** *A directed graph* $G = (V, E)$ *is a* generalized PPAD graph *if* (1) *there exist exactly one vertex* $v_S \in V$ *with* $\Delta_O(v_S) = \Delta_I(v_S) + 1$*, and exactly one* $v_T \in V$ *with* $\Delta_I(v_T) = \Delta_O(v_T)+1$*;* (2) *all vertices in* $V - \{v_S, v_T\}$ *satisfy Euler's condition; and* (3) *if* (v_1, v_2) *is a directed edge in* E*, then* $(v_2, v_1) \notin E$.

We refer to v_S *and* v_T *as the* starting *and* ending *vertices of* G*, respectively.* G *is a* PPAD graph *if in addition* $\Delta_I(v), \Delta_O(v) \leq 1$*, for all* $v \in V$.

Edges of a PPAD graph form a collection of disjoint directed cycles and a directed path from v_S to v_T. In this paper, we are interested in a special family of PPAD graphs over $\mathbb{Z}^d_n$: A directed graph $G = (\mathbb{Z}^d_n, E)$ is a *generalized grid-PPAD graph* over $\mathbb{Z}^d_n$ if it is a generalized PPAD graph and the underlying undirected graph of G is a subgraph of the grid graph defined over $\mathbb{Z}^d_n$. Moreover, if G is also a PPAD graph, then we say G is a *grid-PPAD graph*. G^* and G' in Figure 1 are examples of generalized grid-PPAD graph and grid-PPAD graph, respectively. We now define the query model $\mathbb{B}_G$ for accessing a grid-PPAD graph.

Definition 2.5. *Given a grid-PPAD graph* $G = (\mathbb{Z}^d_n, E)$*, we define* $\mathbb{B}_G : \mathbb{Z}^d_n \to (\{\text{“no”}\} \cup \mathbb{E}^d) \times (\{\text{“no”}\} \cup \mathbb{E}^d)$ *as follows: for* $\mathbf{v} \in \mathbb{Z}^d_n$ (1) $\mathbb{B}_G(\mathbf{v}) = (\text{“no”}, \mathbf{v}_1 - \mathbf{v})$ *if* $\mathbf{v}$ *is the starting vertex of* G *and* $(\mathbf{v}, \mathbf{v}_1) \in E$*;* (2) $\mathbb{B}_G(\mathbf{v}) = (\mathbf{v} - \mathbf{v}_1, \text{“no”})$ *if* $\mathbf{v}$ *is the ending vertex of* G *and* $(\mathbf{v}_1, \mathbf{v}) \in E$*;* (3) $\mathbb{B}_G(\mathbf{v}) = (\mathbf{v} - \mathbf{v}_1, \mathbf{v}_2 - \mathbf{v})$ *if* $(\mathbf{v}_1, \mathbf{v})$ *and* $(\mathbf{v}, \mathbf{v}_2) \in E$*); and* (4) $\mathbb{B}_G(\mathbf{v}) = (\text{“no”}, \text{“no”})$*, otherwise.*

In other words, for each $\mathbf{v} \in \mathbb{Z}^d_n$, $\mathbb{B}_G(\mathbf{v})$ gives the predecessor and successor of $\mathbf{v}$ in G. We will use the property that if $\mathbb{B}_G(\mathbf{v}) = (\mathbf{s}_1, \mathbf{s}_2)$ and $\mathbf{s}_1, \mathbf{s}_2 \in \mathbb{E}^d$, then $\mathbf{s}_1 + \mathbf{s}_2 \neq 0$.

Let GP^d be the search problem: *Given a triple* $(G, n, \mathbf{u})$ *where* G *is a grid-PPAD graph over grid* $\mathbb{Z}^d_n$ *accessible by* $\mathbb{B}_G$ *and* $\mathbf{u}$ *is the starting vertex of* G *satisfying* $u_d = 1$*, find its ending vertex.* We use $\mathsf{RQ}^d_{\mathsf{GP}}(n)$ to denote the randomized query complexity for solving this problem.

3 Reduction Among Search Problems

In this section, we reduce ES^{d-1} to ZP^d by first reducing ES^{d-1} to GP^d and then reducing GP^d to ZP^d (Theorem 3.1 and 3.2 below). Theorem 2.1 then follows from Theorem 2.3. In this section, we will prove Theorem 3.1. A proof of Theorem 3.2 can be found in the full version [10].

Theorem 3.1 (From ES^{d-1} to GP^d)**.** *For all* $d \geq 2$*,*

$$\mathsf{RQ}^{d-1}_{\mathsf{ES}}(n) \leq 4d \cdot \mathsf{RQ}^d_{\mathsf{GP}}(8n + 1).$$

Theorem 3.2 (From GP^d to ZP^d)**.** *For all* $d \geq 2$*,*

$$\mathsf{RQ}^d_{\mathsf{GP}}(n) \leq \mathsf{RQ}^d_{\mathsf{ZP}}(24n + 7).$$

Proof of Theorem 3.1. We define a map $\mathcal{F}_d$ from $\mathbb{Z}^{d-1}$ to $\mathbb{Z}^d$: for $d = 2$, $\mathcal{F}_2(a) = (a, a)$; and for $d > 2$, $\mathcal{F}_d(\mathbf{a}) = (a_1, a_1 + a_2, ..., a_{d-2} + a_{d-1}, a_{d-1})$. We will crucially use

[5]Here the length of S is unknown to the algorithm.

[6]The parameter n is not the length of S, but the size of the alphabet.

Authorized licensed use limited to: Tsinghua University Library. Downloaded on November 24, 2008 at 00:45 from IEEE Xplore. Restrictions apply.

the following nice property of $\mathcal{F}_d$: For any $k \in [1 : d-1]$ and for any $\mathbf{a} \in \mathbb{Z}^{d-1}$, we can uniquely determine the first k and the last k entries of $\mathbf{a}$, respectively, from the first k and the last k entries of $\mathcal{F}_d(\mathbf{a})$.

Let S be a $(d-1)$-non-repeating string over $\mathbb{Z}_n$ of length $m(d-1)$ for some $m \geq 2$, whose $(d-1)^{st}$ symbol is 1. We view S as a sequence of m points $\mathbf{a}_1, \mathbf{a}_2, ..., \mathbf{a}_m$ in $\mathbb{Z}_n^{d-1}$, where $\mathbf{a}_i = a_{i,1}...a_{i,d-1}$, such that,

$$S = a_{1,1}a_{1,2}...a_{1,d-1}...a_{m,1}a_{m,2}...a_{m,d-1}.$$

From S, we will construct a grid-PPAD graph G' in two stages. In the first stage, we construct a *generalized* grid-PPAD graph G^* over $\mathbb{Z}_{2n}^d$ such that

> (**A.1**) Its starting vertex is $\mathbf{u}^* = \mathcal{F}_d(\mathbf{a}_1)$ and its ending vertex is $\mathbf{w}^* = \mathcal{F}_d(\mathbf{a}_m)$; (**A.2**) For each directed edge $(\mathbf{u}, \mathbf{v})$ with $\mathbf{u} - \mathbf{v} \in \mathbb{E}^d$, at most one query to $\mathbb{B}_S$ is needed to determine whether $(\mathbf{u}, \mathbf{v}) \in G^*$ or not.

Suppose $\mathbf{u}$ and $\mathbf{v} \in \mathbb{Z}_{2n}^d$ are two vertices that differ in only one coordinate, say the i^{th} coordinate, and $\mathbf{e} = (\mathbf{v} - \mathbf{u})/|v_i - u_i| \in \mathbb{E}^d$. Let $E(\mathbf{u}, \mathbf{v}) = \{(\mathbf{u}, \mathbf{u}+\mathbf{e}), (\mathbf{u}+\mathbf{e}, \mathbf{u}+2\mathbf{e})...(\mathbf{v}-\mathbf{e}, \mathbf{v})\}$. For $n, m_1, m_2 \in \mathbb{Z}$ and $s \in \{\pm 1\}$, (n, s) is *consistent* with (m_1, m_2) if either *(1)* $m_1 \leq n < m_2$ and $s = 1$; or *(2)* $m_2 < n \leq m_1$ and $s = -1$. Recall a directed path is *simple* if it contains each vertex at most once.

Now consider two consecutive points $\mathbf{a} = \mathbf{a}_t$ and $\mathbf{b} = \mathbf{a}_{t+1}$ in the $(d-1)$-non-repeating string $S = \mathbf{a}_1...\mathbf{a}_m$. We know $\mathbf{a} \neq \mathbf{b}$. We map them to $\mathbf{u} = \mathcal{F}_d(\mathbf{a})$ and $\mathbf{w} = \mathcal{F}_d(\mathbf{b})$ in $\mathbb{Z}_{2n}^d$, and connect them with a path through a sequence of $(d-1)$ vertices $\mathbf{v}_0 = \mathbf{u}, \mathbf{v}_1, ..., \mathbf{v}_{d-1}, \mathbf{v}_d = \mathbf{w}$, where $v_{i,j} = u_j$ if $i < j$ and $v_{i,j} = w_j$ if $i \geq j$.

Note that $\mathbf{v}_{i-1}$ and $\mathbf{v}_i$ differ only in the i^{th} coordinate. Let $P(\mathbf{a}, \mathbf{b}) = \cup_{i=0}^{d-1} E(\mathbf{v}_i, \mathbf{v}_{i+1})$, then $P(\mathbf{a}, \mathbf{b})$ is a *simple* directed path from $\mathbf{u} = \mathbf{v}_0$ to $\mathbf{w} = \mathbf{v}_d$.

Construction of G^* from S: $G^* = (\mathbb{Z}_{2n}^d, \cup_{i=1}^{m-1} P(\mathbf{a}_i, \mathbf{a}_{i+1}))$. See Figure 1 for an example. By Lemma 3.4 below, G^* is a generalized grid-PPAD graph. Property **(A.2)** can be derived from Proposition 3.3.

Proposition 3.3 (Local Characterization of $P(\mathbf{a}, \mathbf{b})$)**.** *Let $\mathbf{a}, \mathbf{b}$ be two points in $\mathbb{Z}_n^{d-1}$. For $\mathbf{v} \in \mathbb{Z}_{2n}^d$ and $s \in \{\pm 1\}$:*

(1) $(\mathbf{v}, \mathbf{v} + s\mathbf{e}_1) \in P(\mathbf{a}, \mathbf{b})$ *iff* (v_1, s) *is consistent with* (a_1, b_1)*,* $a_{d-1} = v_d$*, and* $a_{d-i} = v_{d-i+1} - a_{d-i+1}$ *for all* $i \in [2 : d-1]$*;*

(2) $(\mathbf{v}, \mathbf{v} + s\mathbf{e}_d) \in P(\mathbf{a}, \mathbf{b})$ *iff* (v_d, s) *is consistent with* (a_{d-1}, b_{d-1})*,* $b_1 = v_1$ *and* $b_i = v_i - b_{i-1}$*,* $\forall\, i \in [2 : d-1]$*;*

(3) *When* $k \in [2 : d-1]$*,* $(\mathbf{v}, \mathbf{v} + s\mathbf{e}_k) \in P(\mathbf{a}, \mathbf{b})$ *iff:* (3.0) (v_k, s) *is consistent with* $(a_{k-1} + a_k, b_{k-1} + b_k)$*;* (3.1) $a_{d-1} = v_d$ *and* $a_{d-i} = v_{d-i+1} - a_{d-i+1}$*,* $\forall\, i \in [2 : d-k]$*; and* (3.2) $b_1 = v_1$ *and* $b_i = v_i - b_{i-1}$*,* $\forall\, i \in [2 : k-1]$.

Lemma 3.4 (Structural Correctness)**.** *For all $(d-1)$-non-repeating string $S = \mathbf{a}_1\mathbf{a}_2...\mathbf{a}_m$ over $\mathbb{Z}_n$, if edge $(\mathbf{u}, \mathbf{v}) \in P(\mathbf{a}_i, \mathbf{a}_{i+1})$ then $(\mathbf{u}, \mathbf{v}), (\mathbf{v}, \mathbf{u}) \notin P(\mathbf{a}_j, \mathbf{a}_{j+1})$ for all $j \neq i$.*

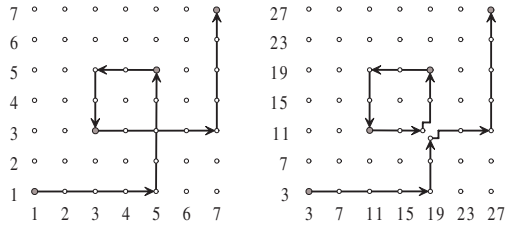

Figure 1. G^* and G' constructed from 1537

Proof. We only prove the case when $\mathbf{e} = \mathbf{v} - \mathbf{u} = s\mathbf{e}_k$ with $1 < k < d$ and $s \in \{\pm 1\}$. The other two cases are similar.

From Proposition 3.3, $(\mathbf{u}, \mathbf{v}) \in P(\mathbf{a}_i, \mathbf{a}_{i+1})$ implies that $\mathbf{a}_i$ and $\mathbf{a}_{i+1}$ satisfy conditions (3.1) and (3.2). If $(\mathbf{u}, \mathbf{v})$ or $(\mathbf{v}, \mathbf{u})$ is in $P(\mathbf{a}_j, \mathbf{a}_{j+1})$, then $\mathbf{a}_j$ and $\mathbf{a}_{j+1}$ also satisfy these two conditions. Thus, $a_{i,k}...a_{i,d-1}a_{i+1,1}...a_{i+1,k-1} = a_{j,k}$ $...a_{j,d-1}a_{j+1,1}...a_{j+1,k-1}$, which contradicts with the assumption that S is $(d-1)$-non-repeating. □

In the second stage, we build a grid-PPAD graph G' over $\mathbb{Z}_{8n+1}^d$ from G^*. Let map $\Gamma(\mathbf{v}) = 4\mathbf{v} - \mathbf{1}$ for all $\mathbf{v} \in \mathbb{Z}_{2n}^d$. Our G', see Figure 1 for an example, satisfies:

> (**B.1**) Its starting vertex is $\mathbf{u}' = \Gamma(\mathbf{u}^*) - 2\mathbf{e}_d$, and its ending vertex $\mathbf{w}'$ satisfies $\|\mathbf{w}' - \Gamma(\mathbf{w}^*)\| \leq 1$; (**B.2**) For each $\mathbf{v} \in \mathbb{Z}_{8n+1}^d$, one can determine $\mathbb{B}_{G'}(\mathbf{v})$ from the predecessors and successors of $\mathbf{u}$ in G^*, where $\mathbf{u} \in \mathbb{Z}_{2n}^d$ is the lexicographically smallest vertex such that $\|\mathbf{v} - \Gamma(\mathbf{u}) \leq 2\|$.

Every vertex in $G^* = (\mathbb{Z}_{2n}^d, E^*)$, other than the starting and ending ones, satisfies Euler's condition. Some vertices may have in-degree and out-degree more than one. In this stage, we systematically unknot high-degree intersections of G^* using a larger space $\mathbb{Z}_{8n+1}^d$.

Two subsets H_1 and H_2 of $\mathbb{E}^d$, where $d \geq 2$, form a *balanced-non-canceling pair* if $|H_1| = |H_2|$ and $\mathbf{s}_1 + \mathbf{s}_2 \neq \mathbf{0}$ for all $\mathbf{s}_1 \in H_1$ and $\mathbf{s}_2 \in H_2$. Let $H_I(\mathbf{u}) = \{\mathbf{e} \in \mathbb{E}^d \,|\, (\mathbf{u} - \mathbf{e}, \mathbf{u}) \in E^*\}$ be the vector differences of $\mathbf{u}$ and its predecessors in G^*. Similarly, let $H_O(\mathbf{u}) = \{\mathbf{e} \in \mathbb{E}^d \,|\, (\mathbf{u}, \mathbf{u}+\mathbf{e}) \in E^*\}$ be the vector differences of the successors of $\mathbf{u}$ and $\mathbf{u}$. In the construction below, we will use the fact that if $\mathbf{u}$ satisfies Euler's condition then (H_I, H_O) is a balanced-non-canceling pair.

Using the procedure of Figure 2, we define a directed graph $G[H_1, H_2] = (\{-1, 0, +1\}^d, E[H_1, H_2])$ for every balanced-non-canceling pair H_1 and H_2. $G[H_1, H_2]$ has the following properties: *(1)* For every vertex $\mathbf{u}$ of $G[H_1, H_2]$, $\Delta_I(\mathbf{u}), \Delta_O(\mathbf{u}) \leq 1$; *(2)* A vertex $\mathbf{u}$ has $\Delta_I(\mathbf{u}) = 0$ and $\Delta_O(\mathbf{u}) = 1$ iff there exists an $\mathbf{e} \in H_1$ such that $\mathbf{u} = -\mathbf{e}$; *(3)* A vertex $\mathbf{u}$ has $\Delta_I(\mathbf{u}) = 1$ and $\Delta_O(\mathbf{u}) = 0$ iff there exists an $\mathbf{e} \in H_2$ such that $\mathbf{u} = +\mathbf{e}$.

Construction of G' from G^*: Let $\mathbf{u}^*$ be the starting vertex and $\mathbf{w}^*$ be the ending vertex of G^*. We build $G' = (\mathbb{Z}_{8n+1}^d, E')$ by applying the procedure of Figure 2 locally

Authorized licensed use limited to: Tsinghua University Library. Downloaded on November 24, 2008 at 00:45 from IEEE Xplore. Restrictions apply.

1: set edge set $E[H_1, H_2] = \emptyset$
2: **while** $H_1 \neq \emptyset$ **do**
3: let $\mathbf{s}_1$ be the smallest vector in H_1 and $\mathbf{s}_2$ be the largest vector in H_2 under the lexicographical ordering
4: set $H_1 = H_1 - \{\mathbf{s}_1\}$ and $H_2 = H_2 - \{\mathbf{s}_2\}$; insert $\{(-\mathbf{s}_1, -\mathbf{s}_1 + \mathbf{s}_2), (-\mathbf{s}_1 + \mathbf{s}_2, \mathbf{s}_2)\}$ into $E[H_1, H_2]$

Figure 2. Construction of Graph $G[H_1, H_2]$

to every vertex $\mathbf{u} \in \mathbb{Z}_{2n}^d$ of G^*. When $\mathbf{u} \in \{\mathbf{u}^*, \mathbf{w}^*\}$, we use a slight modification of $(H_I(\mathbf{u}), H_O(\mathbf{u}))$. Initially, the edge set $E' = \emptyset$. Recall $\Gamma(\mathbf{u}) = 4\mathbf{u} - \mathbf{1}$.

[local embedding of the starting vertex $\mathbf{u}^*$]
As $u_d^* = 1$, $\mathbf{e}_d \notin H_I(\mathbf{u}^*)$ and $-\mathbf{e}_d \notin H_O(\mathbf{u}^*)$. Let $H_I = H_I(\mathbf{u}^*) \cup \{\mathbf{e}_d\}$. Add directed edges $(\Gamma(\mathbf{u}^*) - 2\mathbf{e}_d, \Gamma(\mathbf{u}^*) - \mathbf{e}_d)$ and $(\Gamma(\mathbf{u}^*) + \mathbf{s}_1, \Gamma(\mathbf{u}^*) + \mathbf{s}_2)$ to E' for all edges $(\mathbf{s}_1, \mathbf{s}_2)$ in $G[H_I, H_O(\mathbf{u}^*)]$.

[local embedding of the ending vertex $\mathbf{w}^*$]
As $|H_I(\mathbf{w}^*)| = |H_O(\mathbf{w}^*)| + 1$, $H_I(\mathbf{w}^*) \neq \emptyset$. Let $\mathbf{e}$ be the smallest vector in $H_I(\mathbf{w}^*)$, and $H_I = H_I(\mathbf{w}^*) - \{\mathbf{e}\}$. Add edges $(\Gamma(\mathbf{w}^*) + \mathbf{s}_1, \Gamma(\mathbf{w}^*) + \mathbf{s}_2)$ to E' for all edges $(\mathbf{s}_1, \mathbf{s}_2)$ in $G[H_I, H_O(\mathbf{w}^*)]$.

[local embedding of other vertices $\mathbf{u}$]
For each $\mathbf{u} \in G^*$, add $(\Gamma(\mathbf{u}) + \mathbf{s}_1, \Gamma(\mathbf{u}) + \mathbf{s}_2)$ to E' for all edges $(\mathbf{s}_1, \mathbf{s}_2)$ in $G[H_I(\mathbf{u}), H_O(\mathbf{u})]$.

[connecting local embeddings]
For each $(\mathbf{u}, \mathbf{v}) \in G^*$, letting $\mathbf{e} = \mathbf{v} - \mathbf{u} \in \mathbb{E}^d$, we add $(\Gamma(\mathbf{u}) + \mathbf{e}, \Gamma(\mathbf{u}) + 2\mathbf{e})$, $(\Gamma(\mathbf{u}) + 2\mathbf{e}, \Gamma(\mathbf{u}) + 3\mathbf{e})$ to E'.

It is quite mechanical to check that G' is a grid-PPAD graph that satisfies **(B.1)** and **(B.2)**. Theorem 3.1 follows directly from Property **(A.1)**, **(A.2)**, **(B.1)** and **(B.2)**. □

4 Randomized Lower Bound for ES^d

The technical objective of this section is to construct a distribution $\mathcal{S}$ of d-non-repeating strings and show that, for a random string S drawn according to $\mathcal{S}$, every deterministic algorithm for ES^d needs expected $(\Omega(n))^d$ queries to $\mathbb{B}_S$. Thus, by Yao's Minimax Principle [38], we have $\mathsf{RQ}_{\mathsf{ES}}^d(n) = (\Omega(n))^d$. We will apply random permutations hierarchically to define the distribution $\mathcal{S}$, and ensure that a random string from $\mathcal{S}$ has sufficient entropy that its search problem is expected to be difficult.

4.1 Hierarchical Construction of Random d-Non-Repeating Strings

Let $\mathbb{J}_n = [2 : 2n + 2]$, $\mathbb{O}_n = \{3, 5, 7, ..., 2n + 1\}$, and $\mathbb{F}_n = \{4, 6, ..., 2n + 2\}$. Let strings $S_0 = 2$, $S_1 = 3 \circ 4$ ... $S_n = (2n + 1) \circ (2n + 2)$. Each permutation $\pi : [1 : n] \rightarrow [1 : n]$ defines a string $C = S_0 \circ S_{\pi(1)} \circ ... \circ S_{\pi(n)}$ which we refer to as a *connector* over $\mathbb{J}_n$. Let $r[C] = 2\pi(n) + 2$, the last symbol of C. We use $\phi_C(2)$ to denote the right neighbor of 2. Each $s \in \mathbb{J}_n - \{2, r[C_\pi]\}$ has two neighbors in C. The left neighbor of an even s is $s - 1$, and we use $\phi_C(s)$ to denote its right neighbor; the right neighbor of an odd s is $s + 1$ and we use $\phi_C(s)$ to denote its left neighbor. Clearly, if $\phi_C(s) = t$ then $\phi_C(t) = s$.

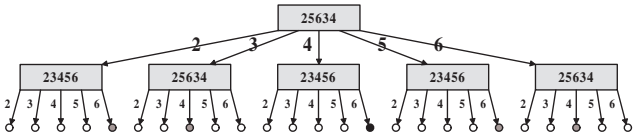

Figure 3. A valid $(2, 2)$-ToC $\mathcal{T}$

Our hierarchical framework is built on $T_{n,d}$, the rooted complete-$(2n + 1)$-nary tree of height d. In $T_{n,d}$, each internal node u is connected to its $(2n + 1)$ children by edges with distinct labels from $\mathbb{J}_n$; if u is connected to v by an edge labeled with j, then we call v the j^{th}-*successor* of u. Each node v of $T_{n,d}$ has a *natural name*, $\mathsf{name}(v)$, the concatenation of labels along the path from the root of $T_{n,d}$ to v. Let $\mathsf{height}(v)$ and $\mathsf{level}(v)$ denote the height and level of node v in the tree. For example, the height of the root is d and the level of the root is 0.

Definition 4.1 (Tree-of-Connectors)**.** *An (n, d)-*ToC *$\mathcal{T}$ is a tree $T_{n,d}$ in which each* internal *node v is associated with a connector C_v over $\mathbb{J}_n$. The $(r[C_v])^{th}$-successor of v is referred to as the* last successor *of v. The* tail *of v, $\mathsf{tail}(v)$, is the leaf reachable from v by last-successor relations. The tail of a leaf is itself. The* tail *of $\mathcal{T}$, $\mathsf{tail}(\mathcal{T})$, is the tail of its root. The* head *of a leaf u, $\mathsf{head}(u)$, is the ancestor of u with the largest height such that u is its tail; if no such ancestor exists, then $\mathsf{head}(u) = u$.*

Definition 4.2 (Valid ToC)**.** *An (n, d)-*ToC *$\mathcal{T}$ is* valid *if for every internal v and for each pair of $s, t \in \mathbb{J}_n$ with $\phi_{C_v}(s) = t$, $\mathsf{name}(u_s)$ and $\mathsf{name}(u_t)$ share a common suffix of length $\mathsf{height}(v) - 1$, where u_s and u_t are the tails of the s^{th}-successor and t^{th}-successor of v, respectively.*

Definition 4.3 ($\mathbb{B}_\mathcal{T}$ for accessing $\mathcal{T}$)**.** *Suppose $\mathcal{T}$ is a valid (n, d)-ToC. The input to $\mathbb{B}_\mathcal{T}$ is a point $\mathbf{q} \in (\mathbb{J}_n)^d$ (defining the name of a leaf u in $\mathcal{T}$). Let $h = \mathsf{height}(\mathsf{head}(u))$. If u is the tail of $\mathcal{T}$, i.e., $h = d$, then $\mathbb{B}_\mathcal{T}(\mathbf{q}) = \mathcal{T}$. Otherwise, let $v_1 = \mathsf{head}(u)$ and let v be the parent of v_1. Note that v_1 is the $(q_{d-h})^{th}$-successor of v. Let $\mathcal{T}_1$ be the tree rooted at v_1. As $u \neq \mathsf{tail}(v)$, $\phi_{C_v}(q_{d-h})$ is defined and let $\mathcal{T}_2$ be the subtree rooted at the $(\phi_{C_v}(q_{d-h}))^{th}$-successor of v. Then, $\mathbb{B}_\mathcal{T}(\mathbf{q}) = (h, \phi_{C_v}(q_{d-h}), \mathcal{T}_1, \mathcal{T}_2)$.*

We define our final search problem Name-the-Tail, NT^d as: *Given a valid (n, d)-ToC $\mathcal{T}^*$ accessible by $\mathbb{B}_{\mathcal{T}^*}$, find the name of its tail.* Theorem 2.3 follows from the next two theorems. We will prove Theorem 4.4 in Section 4.3.

Authorized licensed use limited to: Tsinghua University Library. Downloaded on November 24, 2008 at 00:45 from IEEE Xplore. Restrictions apply.

Theorem 4.4 (Complexity of NT^d)**.** *For all $d \geq 1$ and sufficiently large n, $\mathsf{RQ}^d_{\mathsf{NT}}(n) \geq 2^{-(d+1)}(24^{-d}n)^d$.*

Theorem 4.5 (From NT^d to ES^d)**.** *For all $d \geq 1$, we have $\mathsf{RQ}^d_{\mathsf{NT}}(n) \leq \mathsf{RQ}^d_{\mathsf{ES}}(4n+4)$.*

Proof of Theorem 4.5. For two strings $S_1 = a_1...a_k$ and $S_2 = b_1...b_t$, let $S_1 \circ S_2 = a_1...a_k\, b_1...b_t$. For $d \geq 1$, if $a_{k-d+i} = b_i$ for all $i \in [1:d]$, then let $S_1 \circ_d S_2 = a_1...a_k\, b_{d+1}...b_t$. Given a string S over $\mathbb{Z}$ of length $k \cdot d$, we write S as $\mathbf{u}_1\mathbf{u}_2...\mathbf{u}_k$ with $\mathbf{u}_i \in \mathbb{Z}^d$. For $t \in \mathbb{Z}$, let $\mathsf{insert}_d(S,t) = \mathbf{u}_1 \circ t \circ \mathbf{u}_2 \circ t ... \mathbf{u}_{k-1} \circ t \circ \mathbf{u}_k$.

We need to build a d-non-repeating string from a valid (n,d)-ToC $\mathcal{T}$. In fact, we will construct two strings $S[\mathcal{T}]$ and $Q[\mathcal{T}]$ over $\mathbb{Z}_{4n+4}$. We define $\mathbf{s}_d \in \mathbb{Z}^d$ to be $\mathbf{s}_d = (2, ..., 2, 1)$ ($s_1 = 1$), and $\mathcal{F}(\mathbf{p}) = (2p_1, ..., 2p_{d-1}, 2p_d - 1)$ for $\mathbf{p} \in \mathbb{Z}^d$. Then string $S[\mathcal{T}]$ starts with $\mathbf{s}_d$ and ends with $\mathcal{F}(\mathsf{name}(\mathsf{tail}(\mathcal{T})))$; $Q[\mathcal{T}]$ starts with $\mathcal{F}(\mathsf{name}(\mathsf{tail}(\mathcal{T})))$ and ends with $\mathbf{s}_d$.

We use the following recursive procedure. Let r be the root of $\mathcal{T}$ and $C_r = a_1...a_{2n+1}$ be the connector at r. When $d = 1$, $S[\mathcal{T}] = 1b_1b_2...b_{2n+1}$ and $Q[\mathcal{T}] = b_{2n+1}...b_2b_11$, where $b_i = 2a_i - 1$. When $d \geq 2$,

1. let $\mathcal{T}_i$ be the subtree of $\mathcal{T}$ rooted at the $(a_i)^{th}$-successor of r and let $\mathbf{p}_i \in (\mathbb{F}_n)^{d-1}$ be the name of the tail of $\mathcal{T}_i$ given by $\mathcal{T}_i$ (not by $\mathcal{T}$).
2. for every odd integer $i \in [1:2n+1]$, set string $S'_i = \mathsf{insert}_{d-1}(S[\mathcal{T}_i], 2a_i)$ which starts with $\mathbf{s}_{d-1}$ and ends with $\mathcal{F}(\mathbf{p}_i)$; for every even $i \in [1:2n+1]$, set string $S'_i = \mathsf{insert}_{d-1}(Q[\mathcal{T}_i], 2a_i)$ which starts with $\mathcal{F}(\mathbf{p}_i)$ and ends with $\mathbf{s}_{d-1}$; $S[\mathcal{T}] = \mathbf{s}_d \circ_{d-1} S'_1 \circ_{d-1} S'_2 \circ_{d-1} S'_3 \circ_{d-1} ... \circ_{d-1} S'_{2n} \circ_{d-1} S'_{2n+1}$;
3. for every odd integer $i \in [1:2n+1]$, we set string $Q'_i = \mathsf{insert}_{d-1}(Q[\mathcal{T}_i], 2a_i)$, which starts with $\mathcal{F}(\mathbf{p}_i)$ and ends with $\mathbf{s}_{d-1}$; for every even $i \in [1:2n+1]$, $Q'_i = \mathsf{insert}_{d-1}(S[\mathcal{T}_i], 2a_i)$, which starts with $\mathbf{s}_{d-1}$ and ends with $\mathcal{F}(\mathbf{p}_i)$; $Q[\mathcal{T}] = (2a_{2n+1}) \circ Q'_{2n+1} \circ_{d-1} Q'_{2n} \circ_{d-1} ... \circ_{d-1} Q'_2 \circ_{d-1} Q'_1 \circ \mathbf{s}_d$.

The two strings for the example in Figure 3 are:

$$S[\mathcal{T}] = {}^{2}1^{4}3^{4}5^{4}7^{4}9^{4}11^{10}9^{10}7^{10}5^{10}3^{10}1^{12}3^{12}9^{12}11$$
$$\quad {}^{12}5^{12}7^{6}5^{6}11^{6}9^{6}3^{6}1^{8}3^{8}5^{8}7^{8}9^{8}11$$
$$Q[\mathcal{T}] = {}^{8}11^{8}9^{8}7^{8}5^{8}3^{8}1^{6}3^{6}9^{6}11^{6}5^{6}7^{12}5^{12}11^{12}9^{12}3$$
$$\quad {}^{12}1^{10}3^{10}5^{10}7^{10}9^{10}11^{4}9^{4}7^{4}5^{4}3^{4}1^{2}1$$

The correctness of our construction (and thus, Theorem 4.5) can be established using the next two lemmas [10].

Lemma 4.6 (Non-Repeating)**.** *If $\mathcal{T}$ is a valid (n,d)-ToC, then both $S[\mathcal{T}]$ and $Q[\mathcal{T}]$ are d-non-repeating.*

Lemma 4.7 (Asking $\mathbb{B}_{\mathcal{T}}$)**.** *Suppose $\mathcal{T}$ is a valid (n,d)-ToC, $S = S[\mathcal{T}]$ and $Q = Q[\mathcal{T}]$. For any $\mathbf{u} \in \mathbb{Z}^d_{4n+4}$, we can compute $\mathbb{B}_S(\mathbf{u})$ and $\mathbb{B}_Q(\mathbf{u})$ by querying $\mathbb{B}_{\mathcal{T}}$ at most once.*

4.2 Knowledge Representation in NT^d

An algorithm for NT^d tries to learn about the connectors in $\mathcal{T}^*$ by repeatedly querying its leaves. To capture its intermediate knowledge about this $\mathcal{T}^*$, we introduce a notion of partial connectors.

Let $\sigma = [\sigma(1), ..., \sigma(k)]$ be an array of distinct elements from $\{0, 1, ...n\}$. Then, σ defines a string $S_{\sigma(1)} \circ ... \circ S_{\sigma(k)}$, referred to as a *connecting segment*. Recall $S_0 = 2, S_1 = 3 \circ 4, ...S_n = (2n+1) \circ (2n+2)$. A *partial connector* over $\mathbb{J}_n$ is then a set $\mathcal{C}$ of connecting segments such that *(1)* each $j \in \mathbb{J}_n$ is contained in exactly one segment in $\mathcal{C}$; and *(2)* 2 is the first element of the segment containing it. If $\mathcal{C}$ has $n+1$ segments, that is, $\mathcal{C} = \{2, 3 \circ 4, ..., (2n+1) \circ (2n+2)\}$, then $\mathcal{C}$ is called an *empty connector*. We say a connector C is *consistent* with a partial connector $\mathcal{C}$ if every segment in $\mathcal{C}$ is a substring of C.

Let $r[\mathcal{C}]$ be the last symbol of the segment in $\mathcal{C}$ that starts with 2. Let $L[\mathcal{C}]$ and $R[\mathcal{C}]$, respectively, be the set of first and the last symbols of other segments in $\mathcal{C}$. So, $r[\mathcal{C}] \in \mathbb{F}_n \cup \{2\}$, $L[\mathcal{C}] \subset \mathbb{O}_n$, and $R[\mathcal{C}] \subset \mathbb{F}_n$. Also, $|L[\mathcal{C}]| = |R[\mathcal{C}]|$. If $2 \neq r[\mathcal{C}]$, we use $\phi_{\mathcal{C}}(2)$ to denote its right neighbor. Note that each $s \in \mathbb{J}_n - L[\mathcal{C}] \cup R[\mathcal{C}] \cup \{r[\mathcal{C}], 2\}$ has two neighbors in $\mathcal{C}$. If s is even, we let $\phi_{\mathcal{C}}(s)$ denote its right neighbor and if s is odd, we let $\phi_{\mathcal{C}}(s)$ denote its left neighbor.

Initially, the knowledge of an algorithm for NT^d can be viewed as a tree $\mathcal{T}$ of empty connectors. At each round, the algorithm chooses a query point $\mathbf{q}$ (according to its knowledge about $\mathcal{T}^*$) and asks for $\mathbb{B}_{\mathcal{T}^*}(\mathbf{q})$, which may connect some segments in the partial connectors, and $\mathcal{T}$ is updated. The algorithm succeeds when $\mathcal{T}$ grows into $\mathcal{T}^*$.

At intermediate steps, the knowledge of the algorithm can be expressed by a tree $\mathcal{T}$ of partial connectors. To simplify our proof, we relax the oracle $\mathbb{B}_{\mathcal{T}^*}$ to sometime provide more information to the algorithm than being asked (which will be explained in Section 4.3) so that $\mathcal{T}$ always satisfies the conditions of the following definition:

Definition 4.8 (Valid Tree of Partial Connectors)**.** *$\mathcal{C}$ is a β-*partial connector *for $0 < \beta < 1$ if the number of segments in $\mathcal{C}$ is at least $(1-\beta)n + 1$.*

$\mathcal{T}$ is a valid (n,d,β)-ToPC *if the root of $T_{n,d}$ is associated with a β-partial connector. Moreover, for each internal node $v \in T_{n,d}$ whose children are not leaves, if v has a β-partial connector $\mathcal{C}_v$, then:* (1) *The s^{th}-successor of v, for each $s \in L[\mathcal{C}] \cup R[\mathcal{C}] \cup \{r[\mathcal{C}]\}$, has a β-partial connector;* (2) *For all $s, t \in \mathbb{J}_n$ with $\phi_{\mathcal{C}_v}(s) = t$, the tree $\mathcal{T}_s$ rooted at the s^{th}-successor v_s and the tree $\mathcal{T}_t$ rooted at the t^{th}-successor v_t of v are both valid* ToCs*, and* $\mathsf{name}(\mathsf{tail}(v_s))$ *in $\mathcal{T}_s$ is the same as* $\mathsf{name}(\mathsf{tail}(v_t))$ *in $\mathcal{T}_t$.*

*A valid (n,d)-*ToC *$\mathcal{T}^*$ is* consistent *with a valid (n,d,β)-*ToPC *$\mathcal{T}$, denoted by $\mathcal{T} \models \mathcal{T}^*$, if for each internal node of $T_{n,d}$, its connector in $\mathcal{T}^*$ is consistent with its (partial) connector in $\mathcal{T}$.*

Authorized licensed use limited to: Tsinghua University Library. Downloaded on November 24, 2008 at 00:45 from IEEE Xplore. Restrictions apply.

The definition implies: Given a *valid* (n,d,β)*-ToPC* $\mathcal{T}$, for each internal node v of $T_{n,d}$, it is associated with either a connector or a β-partial connector, and moreover, the subtree of $\mathcal{T}$ rooted at v is either a valid $(n, \mathsf{height}(v))$-ToC or a valid $(n, \mathsf{height}(v), \beta)$-ToPC.

A key to our analysis is Lemma 4.9 below, stating that every valid (n,d,β)-ToPC has a large number of consistent valid (n,d)-ToCs. Moreover, the tails of these valid ToCs are nearly-uniformly distributed. We let set $\mathcal{F}[\mathcal{T}] = \{\mathsf{name}(\mathsf{tail}(\mathcal{T}^*)) \mid \mathcal{T} \models \mathcal{T}^*\}$. Also, for each $\mathbf{p} \in (\mathbb{F}_n)^d$, let $N[\mathcal{T}, \mathbf{p}] = |\{\mathcal{T}^* \mid \mathcal{T} \models \mathcal{T}^*, \mathsf{name}(\mathsf{tail}(\mathcal{T}^*)) = \mathbf{p}\}|$.

Lemma 4.9 (Key Lemma)**.** *For $d \geq 1$ and $\beta \in [0, 24^{-d}]$, $|\mathcal{F}[\mathcal{T}]| \geq ((1-\beta)n)^d$ for each valid (n,d,β)-*ToPC *$\mathcal{T}$. Moreover, for all $\mathbf{p}_1, \mathbf{p}_2 \in \mathcal{F}[\mathcal{T}]$,*

$$\frac{1}{\alpha_d(\beta)} \leq \frac{N[\mathcal{T}, \mathbf{p}_1]}{N[\mathcal{T}, \mathbf{p}_2]} \leq \alpha_d(\beta), \text{ where } \alpha_1(\beta) = 1 \quad (1)$$

$$\text{and } \alpha_d(\beta) = \frac{(\alpha_{d-1}(\beta))^7}{(2(1-\beta)^{d-1} - 1)^3}, \text{ for } d \geq 2. \quad (2)$$

Proof. When $d = 1$, let $\mathcal{C}$ be the only partial connector in $\mathcal{T}$. Clearly, $\mathcal{F}[\mathcal{T}] = R[\mathcal{C}]$. Thus, in this case the lemma is true. We will use this case as the base of the induction.

We will prove by induction on d that both (1) and (**) $|\mathcal{F}[\mathcal{T}]| \geq ((1-\beta)n)^d$ are true for all $d \geq 1$. When $d \geq 2$, let $\mathcal{C} = \{Y_0, Y_1, Y_2, ..., Y_m\}$ be the β-partial connector at the root of $\mathcal{T}$; assume Y_0 is the segment starting with 2. We use r_i and t_i, respectively, to denote the ending and starting symbols of Y_i. For each $k \in \{r_0, r_1 ..., r_m, t_1, ..., t_m\}$, let $\mathcal{T}_k$ denote the $(n, d-1, \beta)$-ToPC at the k^{th}-successor of the root. For each $(i,j) \in [0:m] \times [1:m]$ with $i \neq j$, we let

$$N_{i,j} = \sum_{\mathbf{p} \in \mathcal{F}[\mathcal{T}_{r_i}] \cap \mathcal{F}[\mathcal{T}_{t_j}]} \big(N[\mathcal{T}_{r_i}, \mathbf{p}] \cdot N[\mathcal{T}_{t_j}, \mathbf{p}] \big). \quad (3)$$

Inductively, (1) and (**) hold for $d-1$. Since $\beta \leq 24^{-d} < 24^{-(d-1)}$, we have

$$\big|\mathcal{F}[\mathcal{T}_{r_i}] \cap \mathcal{F}[\mathcal{T}_{t_j}]\big| = \big|\mathcal{F}[\mathcal{T}_{r_i}]\big| + \big|\mathcal{F}[\mathcal{T}_{t_j}]\big| - \big|\mathcal{F}[\mathcal{T}_{r_i}] \cup \mathcal{F}[\mathcal{T}_{t_j}]\big| \geq (2(1-\beta)^{d-1} - 1)\, n^{d-1} > 0,$$

so $N_{i,j} > 0$ for all $(i,j) \in [0:m] \times [1:m]$ with $i \neq j$.

To show (**), it suffices to prove $\mathcal{F}[\mathcal{T}] = \cup_{k \in R[\mathcal{C}]}(k \circ \mathcal{F}[\mathcal{T}_k])$. Clearly, for any $\mathbf{p} \notin \cup_{k \in R[\mathcal{C}]}(k \circ \mathcal{F}[\mathcal{T}_k])$, $N[\mathcal{T}, \mathbf{p}] = 0$. So, let us consider a point $\mathbf{p} \in \cup_{k \in R[\mathcal{C}]}(k \circ \mathcal{F}[\mathcal{T}_k])$. Since $p_1 \in R[\mathcal{C}]$, WLOG, we assume $p_1 = r_m$. We let $\mathcal{P}$ denote the set of permutations $s_0 s_1 ... s_{m-1}$ over $[0 : m-1]$ with $s_0 = 0$. Then

$$N[\mathcal{T}, \mathbf{p}] = \sum_{s_0 ... s_{m-1} \in \mathcal{P}} \Big(\big(\prod_{i=0}^{m-2} N_{s_i, s_{i+1}} \big) \cdot N_{s_{m-1}, m} \cdot N\big[\mathcal{T}_{r_m}, (p_2, p_3, ..., p_d)\big] \Big) > 0, \quad (4)$$

and thus, $\mathcal{F}[\mathcal{T}] = \cup_{k \in R[\mathcal{C}]}(k \circ \mathcal{F}[\mathcal{T}_k])$.

To prove (1), consider $\mathbf{p}_1$ and $\mathbf{p}_2 \in \mathcal{F}[\mathcal{T}]$. There are two basic cases. When $p_{1,1} = p_{2,1}$, Eqn. (1) follows directly from (4) and the inductive hypothesis. When $p_{1,1} \neq p_{2,1}$, WLOG, we assume $p_{1,1} = r_m$ and $p_{2,1} = r_{m-1}$.

Let $\mathcal{P}_1$ denote the set of permutations over $\{0, 1, ..., m-2, m-1\}$ with $s_0 = 0$, and $\mathcal{P}_2$ denote the set of permutations over $\{0, 1, ..., m-2, m\}$ with $s_0 = 0$. For each $P = s_0 s_1 ... s_{m-1} \in \mathcal{P}_1$, let $\Pi(P) \in \mathcal{P}_2$ be the permutation obtained from P by replacing $m-1$ by m. Clearly Π is a bijection from $\mathcal{P}_1$ to $\mathcal{P}_2$.

Now by mimicking Eqn. (4), we can write $N[\mathcal{T}, \mathbf{p}_1]$ and $N[\mathcal{T}, \mathbf{p}_2]$ as two summations: $N[\mathcal{T}, \mathbf{p}_1] = \sum_{P \in \mathcal{P}_1} N_1(P)$, and $N[\mathcal{T}, \mathbf{p}_2] = \sum_{P \in \mathcal{P}_1} N_2(\Pi(P))$, where $N_1(P)$ and $N_2(\Pi(P))$ are given by similar terms as in (4).

We prove for all $P \in \mathcal{P}_1$, $(N_1(P)/N_2(\Pi(P)) \leq \alpha_d(\beta)$, from which Eqn. (1) follows. Let $P = s_0 s_1 ... s_{m-1}$ where $s_k = m-1$ for some $1 \leq k \leq m-1$. For the case when $k < m-1$, after expanding $N_1(P)$ and $N_2(\Pi(P))$, and canceling out the common terms, we get

$$\frac{N_1(P)}{N_2(\Pi(P))} = \frac{N_{s_{k-1}, m-1} \cdot N_{m-1, s_{k+1}}}{N_{s_{k-1}, m} \cdot N_{m, s_{k+1}}} \times \frac{N_{s_{m-1}, m} \cdot N[\mathcal{T}_{r_m}, (p_{1,2}, p_{1,3}, ..., p_{1,d})]}{N_{s_{m-1}, m-1} \cdot N[\mathcal{T}_{r_{m-1}}, (p_{2,2}, p_{2,3}, ..., p_{2,d})]}.$$

We then expand each term $N_{i,j}$ using the definition (Eqn. (3)). It follows from the application of the inductive hypothesis that $N_1(P)/N_2(\Pi(P)) \leq \alpha_d(\beta)$. Similarly, we can establish the same bound for the case $k = m-1$. □

4.3 The Randomized Query Complexity

By querying every leaf, one can solve any instance of NT^d with n^d queries. Below, we prove Theorem 4.4 by showing $\mathsf{RQ}^d_{\mathsf{NT}}(n) = (\Omega(n))^d$. We first relax $\mathbb{B}_{\mathcal{T}^*}$ by extending its domain to $(\mathbb{J}_n)^m$, for all $m \in [1:d]$.

Definition 4.10 (Relaxation of $\mathbb{B}_{\mathcal{T}^*}$)**.** *Suppose $\mathcal{T}^*$ is a valid (n,d)-ToC and $\mathbf{q} \in (\mathbb{J}_n)^m$. Let v be the node such that $\mathsf{name}(v) = q_1 ... q_m$. Let $\mathbf{q}' = \mathsf{name}(\mathsf{tail}(v)) \in (\mathbb{J}_n)^d$ (in $\mathcal{T}^*$). Then, $\mathbb{B}_{\mathcal{T}^*}(\mathbf{q}) = \mathbb{B}_{\mathcal{T}^*}(\mathbf{q}')$.*

Proof (of Theorem 4.4). We consider the distribution $\mathcal{D}$ in which each valid (n,d)-ToC $\mathcal{T}^*$ is chosen with the same probability. We will prove that the expected query complexity of any deterministic algorithm $\mathcal{A}$ for NT^d over $\mathcal{D}$ is $(\Omega(n))^d$. Let $\beta_d = 24^{-d}$. To simplify the proof, we assume n is a multiple of $2 \cdot 24^d$ so that $\beta_d n/2 \in \mathbb{Z}$.

Suppose, at a particular step, the knowledge of $\mathcal{A}$ can be expressed by a valid (n,d,β_d)-ToPC $\mathcal{T} \models \mathcal{T}^*$, which is clearly true initially, and $\mathcal{A}$ wants to query $\mathbf{q} \in (\mathbb{J}_n)^d$. Let u_0 be the root of $\mathcal{T}$ and u_i be the node with $\mathsf{name}(u_i) = q_1 ... q_i$. Let $\mathcal{C}_i$ be the connector or β_d-partial connector at u_i in $\mathcal{T}$ and $\mathcal{T}_i$ be the subtree of $\mathcal{T}$ rooted at u_i. There are

Authorized licensed use limited to: Tsinghua University Library. Downloaded on November 24, 2008 at 00:45 from IEEE Xplore. Restrictions apply.

Query-and-Update$(\mathcal{T}, \mathbf{q})$**, where** $\mathbf{q} \in (\mathbb{J}_n)^d$

1: **if** $\exists\, 0 \le i \le d-1 : \big|R[\mathcal{C}_i]\big| = (1-\beta_d)n$ **then**
2: set m be the smallest of such i ($m \in [0 : d-1]$)
3: **else** set $m = d$
4: **if** $m = 0$ **then** set $\mathcal{T} = \mathcal{T}^*$ **{** set $I = 1$ **}**
5: **else Update** $(\mathcal{T}, (q_1, q_2, ..., q_m), m)$

Update$(\mathcal{T}, \mathbf{q}, m)$**, where** $\mathbf{q} \in (\mathbb{J}_n)^m$ **and** $1 \le m \le d$

6: fetch $\mathbb{B}_{\mathcal{T}^*}(\mathbf{q})$
{ set $A_m = A_m + 1$, $\mathcal{B}_m[A_m] = 0$, $\mathcal{B}_{m,k}[A_m] = 0$ **}**
7: **if** $\mathbb{B}_{\mathcal{T}^*}(\mathbf{q}) = \mathcal{T}^*$ **then** set $\mathcal{T} = \mathcal{T}^*$ **{** set $\mathcal{B}_m[A_m] = 1$ **}**
8: **else** [let $\mathbb{B}_{\mathcal{T}^*}(\mathbf{q}) = (h, r, \mathcal{T}', \mathcal{T}'')$, $m' = d - h - 1$]
9: $\exists\, Y_1, Y_2 \in \mathcal{C}_{m'}$ such that, { the ending symbol of Y_1, the starting symbol of Y_2 } = { $q_{m'+1}, r$ }
10: replace Y_1 and Y_2 in $\mathcal{C}_{m'}$ by the concatenation of Y_1 and Y_2 **{** set $\mathcal{B}_{m,m'}[A_m] = 1$ **}**
11: replace the subtree of $\mathcal{T}$ rooted at $u_{m'+1}$ with $\mathcal{T}'$;
12: replace the subtree of $\mathcal{T}$ rooted at the r-successor of $u_{m'}$ with $\mathcal{T}''$

Figure 4. The Query-and-Update procedure

two cases *(1)* $\forall\, i \in [0, d-1]$, $\mathcal{C}_i$ is a β_d-partial connector and $q_{i+1} \in L[\mathcal{C}_i] \cup R[\mathcal{C}_i] \cup \{r[\mathcal{C}_i]\}$; *(2)* otherwise. By the definition of $\mathbb{B}_{\mathcal{T}^*}$, we can show that in case *(2)*, $\mathbb{B}_{\mathcal{T}^*}(\mathbf{q})$ can be answered based on $\mathcal{T}$ only. So, WLOG, we assume $\mathcal{A}$ never asks queries of the second type.

In case *(1)*, $\mathcal{C}_i$ is a β_d-partial connector for all $i \in [0 : d-1]$. Let $h = \mathsf{height}(\mathsf{head}(\mathbf{q}))$. If $h = d$, then $\mathcal{A}$ gets $\mathcal{T}^*$. Otherwise, the knowledge gained by querying $\mathbb{B}_{\mathcal{T}^*}(\mathbf{q})$ connects two segments in $\mathcal{C}_{d-h-1}$ and replaces the two involved subtrees by the corresponding ones in the 4-tuple $\mathbb{B}_{\mathcal{T}^*}(\mathbf{q})$. The resulting tree $\mathcal{T}$, however, may no longer be a *valid* (n, d, β_d)*-ToPC*, if $|R[\mathcal{C}_{d-h-1}]| = (1-\beta_d)n$ (or the number of segments in $\mathcal{C}_{d-h-1}$ is exactly $(1-\beta_d)n + 1$) before the query. When this happens, we will relax $\mathbb{B}_{\mathcal{T}^*}$ to provide $\mathcal{A}$ more information to ensure that the resulting $\mathcal{T}$ remains a valid (n, d, β_d)-ToPC.

To this end, we consider two subcases when $\mathcal{A}$ queries $\mathbf{q} \in (\mathbb{J}_n)^d$: Case **(1.a)** $\forall\, i \in [0 : d-1]$, $|R[\mathcal{C}_i]| > (1-\beta_d)n$, then $\mathcal{A}$ receives $\mathbb{B}_{\mathcal{T}^*}(\mathbf{q})$ as it requested; Case **(1.b)** $\exists\, i \in [0 : d-1]$ such that $|R[\mathcal{C}_i]| = (1-\beta_d)n$, then we assume $m = \min\{\, i : |R[\mathcal{C}_i]| = (1-\beta_d)n \,\}$. Let $\mathbf{q}' = (q_1, ..., q_m)$. Instead of getting $\mathbb{B}_{\mathcal{T}^*}(\mathbf{q})$, $\mathcal{A}$ gets $\mathbb{B}_{\mathcal{T}^*}(\mathbf{q}')$. In this way, the resulting $\mathcal{T}$ remains a valid (n, d, β_d)-ToPC (or grows into $\mathcal{T}^*$ and $\mathcal{A}$ stops) after the query. Details of the query-and-update procedure can be found in Figure 4.

We introduce some "analysis variables" to aid our analysis. These variables include: *(1)* $I \in \{0, 1\}$: Initially, $I = 0$. *(2)* For each $m \in [1 : d]$, $A_m \in \mathbb{Z}$, and a set of binary sequences $\mathcal{B}_m[...]$ and $\mathcal{B}_{m,k}[...]$ of length A_m, $\forall\, k \in [0 : m-1]$. Initially, $A_m = 0$, and $\mathcal{B}_m$, $\mathcal{B}_{m,k}$ are empty.

In each step, the variables are updated as follows: **(I)** If $m = 0$ in case **(1.b)**, then we set $I = 1$; **(II)** Otherwise, (to unify the discussion in this part, if we have case **(1.a)**, then set $m = d$ and $\mathbf{q}' = \mathbf{q}$) we increase A_m by 1; **(II.1)** if $\mathbb{B}_{\mathcal{T}^*}(\mathbf{q}') = \mathcal{T}^*$, we set $\mathcal{B}_m[A_m] = 1$ and $\mathcal{B}_{m,k}[A_m] = 0$, $\forall\, k \in [0 : m-1]$; **(II.2)** otherwise, if the first component of $\mathbb{B}_{\mathcal{T}^*}(\mathbf{q}')$ is $d - l$, for $l \in [1 : m]$, then $\mathcal{B}_{m,l-1}[A_m] = 1$, $\mathcal{B}_m[A_m] = \mathcal{B}_{m,k}[A_m] = 0$, $\forall\, k : 0 \le k \ne l-1 \le m-1$. Details can also be found in Figure 4 (highlighted in **{}**).

Let $M_i = (\beta_d n/2)^i$, $\forall\, i \in [1 : d]$. Given a valid (n, d)-ToC $\mathcal{T}^*$, if $\mathcal{A}$ stops before making M_d queries, then we let $\{I, A_m, \mathcal{B}_m, \mathcal{B}_{m,k}\}$ be the variables assigned when $\mathcal{A}$ stops; otherwise, $\{I, A_m, \mathcal{B}_m, \mathcal{B}_{m,k}\}$ is assigned after $\mathcal{A}$ makes M_d queries. Furthermore, we define a set of binary strings $\{\overline{\mathcal{B}}_m[1...M_m], \overline{\mathcal{B}}_{m,k}[1...M_m], 1 \le m \le d, 0 \le k \le m-1\}$ from $\{\mathcal{B}_m, \mathcal{B}_{m,k}\}$: for each $1 \le i \le M_m$, $\overline{\mathcal{B}}_m[i] = \mathcal{B}_m[i]$ and $\overline{\mathcal{B}}_{m,k}[i] = \mathcal{B}_{m,k}[i]$ when $i \le \min(A_m, M_m)$; $\overline{\mathcal{B}}_m[i] = \overline{\mathcal{B}}_{m,k}[i] = 0$ when $A_m < i \le M_m$.

As $\mathcal{T}^*$ is chosen randomly from valid (n, d)-ToCs, $\{I, A_m, \overline{\mathcal{B}}_m, \overline{\mathcal{B}}_{m,k}\}$ is a set of random variables defined by the deterministic algorithm $\mathcal{A}$. To assist the analysis, we introduce the following definition. Lemma 4.12 is an important step in our analysis.

Definition 4.11 (c-Biased Distributions)**.** *Suppose we have a probabilistic distribution over strings* $\{0,1\}^m$*. For every binary string* S *of length at most* m*, we define set* $U_S = \{\, S' \in \{0,1\}^m \,\big|\, S \text{ is a prefix of } S' \,\}$*. For* $0 \le c \le 1$*, the distribution is said to be* c-biased *if we have* $\mathsf{Pr}[U_1] \le c$ *and* $\mathsf{Pr}[U_{S\circ 1}] \le c \cdot \mathsf{Pr}[U_S]$ *for all* S *with* $1 \le |S| \le m-1$*.*

Lemma 4.12. *For all* $1 \le m \le d$*, the distribution over* $\overline{\mathcal{B}}_m$ *is* $2/n^m$*-biased; For* $2 \le m \le d$ *and* $0 \le k \le m-2$*, the distribution over* $\overline{\mathcal{B}}_{m,k}$ *is* $2/n^{m-k-1}$*-biased.*

The lemma follows directly from Corollary 4.13 below of our Key Lemma (4.9).

Corollary 4.13. *For* $d \ge 1$ *and* $\beta \in [0, 24^{-d}]$*, let* $\mathcal{T}$ *be a valid* (n, d, β)*-ToPC, and* $N = \sum_{\mathbf{p} \in \mathcal{F}[\mathcal{T}]} N[\mathcal{T}, \mathbf{p}]$ *be the number of consistent ToCs.*

For $\mathbf{q} \in (\mathbb{J}_n)^m$ *where* $m \in [1 : d]$*, let* u_0 *be the root of* $\mathcal{T}$ *and* u_i *be the node with* $\mathsf{name}(u_i) = q_1...q_i$*. Let* $\mathcal{C}_i$ *be the connector or* β_d*-partial connector at* u_i *and* $\mathcal{T}_i$ *be the subtree of* $\mathcal{T}$ *rooted at* u_i*. If* $\forall\, i \in [0, m-1]$*,* $\mathcal{C}_i$ *is a* β_d*-partial connector and* $q_{i+1} \in L[\mathcal{C}_i] \cup R[\mathcal{C}_i] \cup \{r[\mathcal{C}_i]\}$*, then* **(1)** $(N^*/N) \le (2/n^m)$ *where* $N^* = |\{\mathcal{T}' \,\big|\, \mathbb{B}_{\mathcal{T}'}(\mathbf{q}) = \mathcal{T}' \text{ and } \mathcal{T} \models \mathcal{T}'\}|$*; and* **(2)** $(N_k/N) \le 2/n^{m-k-1}$ *where for* $0 \le k \le m-2$*,* N_k *is the number of consistent ToCs* $\mathcal{T}'$ *such that the first component of* $\mathbb{B}_{\mathcal{T}'}(\mathbf{q})$ *is* $d-k-1$*.*

Proof. For each $k \in [0 : m-1]$, let $W_k = \{\mathbf{p} \in \mathcal{F}[\mathcal{T}_k] \subset (\mathbb{F}_n)^{d-k}$ such that $p_i = q_{k+i}, \forall\, i \in [1 : m-k]\}$. Clearly, $|W_k| \le n^{d-m}$. By Lemma 4.9, we have

Authorized licensed use limited to: Tsinghua University Library. Downloaded on November 24, 2008 at 00:45 from IEEE Xplore. Restrictions apply.

$$\frac{N^*}{N} = \frac{\sum_{\mathbf{p}\in W_0} N[\mathcal{T},\mathbf{p}]}{\sum_{\mathbf{p}\in\mathcal{F}[\mathcal{T}]} N[\mathcal{T},\mathbf{p}]} \le \frac{\alpha_d(\beta)\cdot|W_0|}{|\mathcal{F}[\mathcal{T}]|} \le \frac{2}{n^m}.$$

The third inequality uses Proposition A.3 in Appendix A.

To prove the second statement, for $k \in [0 : m-2]$, we consider any connector C^* over $\mathbb{J}_n$ that is consistent with $\mathcal{C}_k$ and satisfies $\phi_{C^*}(q_{k+1}) \neq$ "no" (that is, $r[C^*] \neq q_{k+1}$). Assume $\phi_{C^*}(q_{k+1}) = r$. We let $\mathcal{T}'$ denote the subtree of $\mathcal{T}$ rooted at the r^{th}-successor of u_k. Since $q_{k+1} \in L[\mathcal{C}_k] \cup R[\mathcal{C}_k] \cup \{r[\mathcal{C}_k]\}$, both $\mathcal{T}_{k+1}$ and $\mathcal{T}'$ are $(n, d-k-1, \beta)$-ToPCs. Therefore, we have

$$\frac{\sum_{\mathbf{p}\in W_{k+1}\cap\mathcal{F}[\mathcal{T}_{k+1}]\cap\mathcal{F}[\mathcal{T}']} N[\mathcal{T}_{k+1},\mathbf{p}]\cdot N[\mathcal{T}',\mathbf{p}]}{\sum_{\mathbf{p}\in\mathcal{F}[\mathcal{T}_{k+1}]\cap\mathcal{F}[\mathcal{T}']} N[\mathcal{T}_{k+1},\mathbf{p}]\cdot N[\mathcal{T}',\mathbf{p}]}$$
$$\le \frac{(\alpha_{d-k-1}(\beta))^2\cdot n^{d-m}}{(2(1-\beta)^{d-k-1}-1)\cdot n^{d-k-1}} \le \frac{2}{n^{m-k-1}}. \qquad \square$$

Let B_m, $B_{m,k}$, $\overline{B}_m$ and $\overline{B}_{m,k}$ denote the number of 1's in $\mathcal{B}_m$, $\mathcal{B}_{m,k}$, $\overline{\mathcal{B}}_m$ and $\overline{\mathcal{B}}_{m,k}$, respectively. Then,

Lemma 4.14. *For all $m \in [1:d]$ and $k \in [0:m-2]$, we have $\mathsf{Pr}_{\mathcal{D}}[\overline{B}_m > 0] < 1/(2d^2)$ and*

$$\mathsf{Pr}_{\mathcal{D}}[\overline{B}_{m,k} > \frac{16\cdot M_m}{n^{m-k-1}}] < \frac{1}{2d^2}.$$

Proof. We will use the following fact: Let $\mathcal{D}^m_{IND}$ denote the distribution over $\{0,1\}^m$ where each bit of the string is chosen independently and is equal to 1 with probability c. For any c-biased distribution $\mathcal{D}^m$ over $\{0,1\}^m$ and integer $k \in [1:m]$, we have $\mathsf{Pr}_{S\leftarrow\mathcal{D}^m}[S \text{ has at least } k \text{ 1's}] \le \mathsf{Pr}_{S\leftarrow\mathcal{D}^m_{IND}}[S \text{ has at least } k \text{ 1's}]$.

By Lemma 4.12, $\mathsf{Pr}_{\mathcal{D}}[\overline{B}_m > 0] \le 1-(1-2n^{-m})^{M_m} \le 4(\beta_d/2)^m \le 1/2d^2$. The second inequality uses Propositions A.2 and A.1, and the last inequality uses $\beta_d = 24^{-d}$ and the fact $m \ge 1$. We can apply Lemma 4.12 and the Chernoff bound [11] to prove the second statement. $\square$

Let $[A]$ denote the event that A is true. Let NOT-YET-FOUND$(\mathcal{T}^*)$ denote the event that algorithm $\mathcal{A}$ has not found the tail of $\mathcal{T}^*$ after making $M_d = (\beta_d n/2)^d$ queries. Then [NOT-YET-FOUND $(\mathcal{T}^*)$] $\Longleftrightarrow$ [$I = 0$ and $B_m = 0$, $\forall\, m \in [1:d]$]. The theorem directly follows from Lemma 4.15 below. $\square$

Lemma 4.15. *Let A denote the following event:*

$$\big(\overline{B}_m = 0 \text{ and } \overline{B}_{m,m-1} \le M_m \text{ and}$$
$$\overline{B}_{m,k} \le \frac{16\cdot M_m}{n^{m-k-1}}, \forall\, m \in [1:d], k \in [0, m-2]\big),$$

then we have **(E.1)** $[A]$ *implies* [NOT-YET-FOUND $(\mathcal{T}^*)$] *and* **(E.2)** $\mathsf{Pr}_{\mathcal{D}}[A] \ge 1/2$.

Proof. To prove **(E.1)**, we need the following inequalities which follow from the definition of our analysis variables.

$$A_m \le \tfrac{1}{\beta_d n}\textstyle\sum_{i=m+1}^{d} B_{i,m},\ \forall\, 1 \le m \le d-1; \quad (5)$$
$$I = 1 \implies \textstyle\sum_{i=1}^{d} B_{i,0} \ge \beta_d n. \quad (6)$$

To prove **(E.1)**, it suffices to show that $[A] \Rightarrow [I = 0]$ and $[A] \Rightarrow [B_m = 0,\ \forall\, m \in [1:d]]$. Here we use $[A] \Rightarrow [B]$ to denote if event A is true then event B is true. It follows immediately from the definitions of B_m and $\overline{B}_m$, that if $A_m \le M_m$, then $B_m = \overline{B}_m$. So, we first inductively prove that $[A] \Rightarrow [A_{d-m} \le M_{d-m}, \forall\, m \in [0:d-1]]$.

The base case when $m = 0$ is trivial, since A_d is at most M_d, the total number of queries. We now consider the case when $m \ge 1$, and assume inductively, that $A_i \le M_i$ for all $i \in [d-m+1:d]$. Consequently, for all $i \in [d-m+1:d]$ and $j \in [0, i-1]$, $\overline{B}_i = B_i$ and $\overline{B}_{i,j} = B_{i,j}$. By Eqn. (5),

$$A_{d-m} \le \sum_{i=d-m+1}^{d} \frac{B_{i,d-m}}{\beta_d n} = \sum_{i=d-m+1}^{d} \frac{\overline{B}_{i,d-m}}{\beta_d n}$$
$$\le \frac{1}{\beta_d n}\Big(M_{d-m+1} + \sum_{i=d-m+2}^{d}\Big(\frac{16\cdot M_i}{n^{i-d+m-1}}\Big)\Big)$$
$$\le M_{d-m}\Big(\frac{1}{2} + 8\sum_{i=d-m+2}^{d}\Big(\frac{\beta_d}{2}\Big)^{i-d+m-1}\Big) < M_{d-m}.$$

Thus, $[A] \Rightarrow [B_m = 0,\ \forall\, m \in [1:d]]$. Now we prove $[A]$ implies $[I = 0]$. By Eqn. (5), it suffices to show $[A]$ implies $[\sum_{m=1}^{d} B_{m,0} < \beta_d n]$. Assume $[A]$, then

$$\sum_{i=m}^{d} B_{m,0} = \overline{B}_{1,0} + \sum_{m=2}^{d}\overline{B}_{m,0} \le M_1 + \sum_{m=2}^{d}\frac{16\cdot M_m}{n^{m-1}}$$
$$= \beta_d n\Big(\frac{1}{2} + 8\sum_{m=2}^{d}\big(\frac{\beta_d}{2}\big)^{m-1}\Big) < \beta_d n.$$

The first equation follows from $[A] \Rightarrow [A_{d-m} \le M_{d-m}, \forall m \in [0:d-1]]$ and the first inequality uses $\overline{B}_{m,m-1} \le M_m$ for all $m \in [1:d]$. Finally, by Lemma 4.14, we have

$$\mathsf{Pr}_{\mathcal{D}}[A] \ge 1 - \Big(\sum_{m=1}^{d}\mathsf{Pr}_{\mathcal{D}}\big[\overline{B}_m > 0\big] + \sum_{m=1}^{d}\sum_{k=1}^{m-2}\mathsf{Pr}_{\mathcal{D}}\Big[\overline{B}_{m,k} > \frac{16\cdot M_m}{n^{m-k-1}}\Big]\Big) \ge \frac{1}{2}.$$

$\square$

5 Open Problems

We conjecture that FPC is also strictly harder than local search in the quantum query model over $[1:n]^d$.

Our current lower bound does not apply to the case when n is small and d is large. For example, one can define the END-OF-A-PATH problem over hypercubes $\{0,1\}^n$. Both the randomized and quantum query complexity of this problem remains open.

We conclude this paper with the following conjecture: If **PPAD** is in **FP**, then **PLS** is in **FP**.

Authorized licensed use limited to: Tsinghua University Library. Downloaded on November 24, 2008 at 00:45 from IEEE Xplore. Restrictions apply.

6 Acknowledgments

We would like to thank Dan Spielman, Xiaoming Sun, Xiaotie Deng, Kyle Burke and Stan Sclaroff for discussions that are invaluable to this work.

References

[1] S. Aaronson. Lower bounds for local search by quantum arguments. In *STOC 2004*, pages 465–474.

[2] D. Aldous. Minimization algorithms and random walk on the d-cube. *Annals of Probability*, 11(2):403–413, 1983.

[3] K. Arrow and G. Debreu. Existence of an equilibrium for a competitive economy. *Econometrica*, 22(3):265–290, 1954.

[4] P. Beame, S. Cook, J. Edmonds, R. Impagliazzo, and T. Pitassi. The relative complexity of NP search problems. *J. Comput. Syst. Sci.*, 57:3–19, 1998.

[5] L. Brouwer. Über abbildung von mannigfaltigkeiten. *Mathematische Annalen*, 71:97–115, 1910.

[6] X. Chen and X. Deng. On algorithms for discrete and approximate Brouwer fixed points. In *STOC 2005*.

[7] X. Chen and X. Deng. On the complexity of 2d discrete fixed point problem. In *ICALP 2006*, pages 489–500.

[8] X. Chen and X. Deng. Settling the complexity of two-player Nash equilibrium. In *FOCS 2006*.

[9] X. Chen, X. Deng, and S.-H. Teng. Computing Nash equilibria: Approximation and smoothed complexity. In *FOCS 2006*.

[10] X. Chen and S.-H. Teng. Paths beyond local search: A nearly tight bound for randomized fixed-point computation. *arXiv:cs/0702088*, 2007.

[11] H. Chernoff. Asymptotic efficiency for tests based on the sum of observations. *Ann. Math. Stat.*, 23:493–507, 1952.

[12] B. Codenotti, A. Saberi, K. Varadarajan, and Y. Ye. Leontief economies encode nonzero sum two-player games. *ECCC, TR05-055*, 2005.

[13] G. Dantzig. Maximization of linear function of variables subject to linear inequalities. In T. Koopmans, editor, *Activity Analysis of Production and Allocation*, pages 339–347.

[14] C. Daskalakis, P. Goldberg, and C. Papadimitriou. The complexity of computing a Nash equilibrium. In *STOC 2006*.

[15] X. Deng, C. Papadimitriou, and S. Safra. On the complexity of price equilibria. *Journal of Computer and System Sciences*, 67(2):311–324, 2003.

[16] H. Edelsbrunner. *Algorithms in Combinatorial Geometry*. Springer-Verlag New York, Inc., 1987.

[17] K. Friedl, G. Ivanyos, M. Santha, and F. Verhoeven. On the black-box complexity of Sperner's lemma. In *15th FCT*, pages 245–257, 2005.

[18] M. Hirsch, C. Papadimitriou, and S. Vavasis. Exponential lower bounds for finding Brouwer fixed points. *Journal of Complexity*, 5:379–416, 1989.

[19] L.-S. Huang and S.-H. Teng. On the approximation and smoothed complexity of Leontief market equilibria. *ECCC, TR06-031*, 2006.

[20] T. Iimura, K. Murota, and A. Tamura. Discrete fixed point theorem reconsidered. *Journal of Mathematical Economics*, 41:1030–1036, 2005.

[21] D. Johnson, C. Papadimtriou, and M. Yannakakis. How easy is local search? *J. Comput. Syst. Sci.*, 37(1):79–100, 1988.

[22] N. Karmarkar. A new polynomial time algorithm for linear programming. *Combinatorica*, 4:373–395, 1984.

[23] V. Klee and G. Minty. How good is the simplex algorithm? In O. Shisha, editor, *Inequalities – III*, pages 159–175. Academic Press, 1972.

[24] C. Lemke and J. J.T. Howson. Equilibrium points of bimatrix games. *J. Soc. Indust. Appl. Math.*, 12:413–423, 1964.

[25] J. Nash. Equilibrium point in n-person games. *Porceedings of the National Academy of the USA*, 36(1):48–49, 1950.

[26] Y. Nesterov and A. Nemirovskii. *Interior Point Polynomial Algorithms in Convex Programming*, volume 13 of *Studies in Applied Mathematics*. SIAM, 1993.

[27] J. Orlin, A. Punnen, and A. Schulz. Approximate local search in combinatorial optimization. In *SODA 2004*.

[28] C. Papadimitriou. On inefficient proofs of existence and complexity classes. In *Proceedings of the 4th Czechoslovakian Symposium on Combinatorics*, 1991.

[29] C. Papadimitriou. On the complexity of the parity argument and other inefficient proofs of existence. *Journal of Computer and System Sciences*, pages 498–532, 1994.

[30] M. Santha and M. Szegedy. Quantum and classical query complexities of local search are polynomially related. In *STOC 2004*, pages 494–501.

[31] R. Savani and B. von Stengel. Exponentially many steps for finding a nash equilibrium in a bimatrix game. In *FOCS 2004*, pages 258–267.

[32] H. Scarf. The approximation of fixed points of a continuous mapping. *SIAM Journal on Applied Mathematics*, 15:997–1007, 1967.

[33] H. Scarf. On the computation of equilibrium prices. In W. Fellner, editor, *Ten Economic Studies in the Tradition of Irving Fisher*. New York: John Wiley & Sons, 1967.

[34] H. Scarf. *The Computation of Economic Equilibria*. Yale University Press, 1973.

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

[36] D. Spielman and S.-H. Teng. Smoothed analysis of algorithms: Why the simplex algorithm usually takes polynomial time. *Journal of ACM*, 51(3):385–463, 2004.

[37] X. Sun and A.-C. Yao. On the quantum query complexity of local search in two and three dimensions. In *FOCS 2006*, pages 429–438.

[38] A.-C. Yao. Probabilistic computations: Towards a unified measure of complexity. In *FOCS 1977*, pages 222–227.

[39] S. Zhang. New upper and lower bounds for randomized and quantum local search. In *STOC 2006*, pages 634–643.

A Inequalities

Proposition A.1. *For all* $\beta \geq 0$, $1 - \beta \leq e^{-\beta}$.

Proposition A.2. *For all* $0 \leq \beta \leq 1/3$, $1 - \beta \geq e^{-2\beta}$.

Lemma A.3. *For all* $d \geq 1$ *and* $\beta \in [0, 24^{-d}]$, $\alpha_d(\beta) \leq e^{2 \cdot 24^{d-1}\beta}$.

Proof. This lemma can be proved by induction on d. □

Authorized licensed use limited to: Tsinghua University Library. Downloaded on November 24, 2008 at 00:45 from IEEE Xplore. Restrictions apply.