Lower Bounds for Local Monotonicity Reconstruction from Transitive-Closure Spanners

Arnab Bhattacharyya[1⋆], Elena Grigorescu[1⋆], Madhav Jha[2⋆⋆], Kyomin Jung[3], Sofya Raskhodnikova[2⋆⋆], and David P. Woodruff[4]

[1] Massachusetts Institute of Technology, USA. {abhatt,elena_g}@mit.edu.
[2] Pennsylvania State University, USA. {mxj201,sofya}@cse.psu.edu.
[3] Korea Advanced Institute of Science and Technology, Korea. kyomin@kaist.edu.
[4] IBM Almaden Research Center, USA. dpwoodru@us.ibm.com.

Abstract. Given a directed acyclic graph (DAG) $G_n = (V_n, E)$, a function *on* G_n is given by $f : V_n \to \mathbb{R}$. Such a function is *monotone* if $f(x) \leq f(y)$ for all $(x, y) \in E$. A *local monotonicity reconstructor* for G_n, introduced by Saks and Seshadhri (SICOMP 2010), is a randomized algorithm that, given access to an oracle for an almost monotone function $f : V_n \to \mathbb{R}$ *on* G_n, can quickly evaluate a related function $g : V_n \to \mathbb{R}$ which is guaranteed to be monotone. Furthermore, the reconstructor can be implemented in a distributed manner.

Given a directed graph $G = (V, E)$ and an integer $k \geq 1$, a *k-transitive-closure-spanner (k-TC-spanner)* of G is a directed graph $H = (V, E_H)$ that has (1) the same transitive-closure as G and (2) diameter at most k. Transitive-closure spanners are a common abstraction for applications in access control, property testing and data structures.

In this paper, we show a connection between 2-TC-spanners of G_n and local monotonicity reconstructors for G_n. We show that an efficient local monotonicity reconstructor for G_n implies a sparse 2-TC-spanner of G_n, providing a new technique for proving lower bounds for local monotonicity reconstructors. We present tight upper and lower bounds on the size of the sparsest 2-TC-spanners of the directed hypercube and hypergrid, DAGs which are very-well studied in this area. These bounds imply lower bounds for local monotonicity reconstructors for the hypergrid (hypercube) that nearly match the known upper bounds.

Keywords: Property Testing, Property Reconstruction, Monotone Functions, Spanners, Hypercube, Hypergrid

---

⋆ A.B. is supported by a DOE Computational Science Graduate Fellowship and NSF Awards 0514771, 0728645, 0732334. E.G. is supported by NSF award CCR-0829672.
⋆⋆ Supported by NSF/CCF award 0729171. S.R. is also supported by NSF/CCF CAREER award 0845701.

1 Introduction

Graph spanners were introduced in the context of distributed computing [1], and since then have found numerous applications, such as efficient routing [2–6], simulating synchronized protocols in unsynchronized networks [7], parallel and distributed algorithms for approximating shortest paths [8–10], and algorithms for distance oracles [11, 12]. Several variants on graph spanners have been defined. In this work, we focus on *transitive-closure* spanners that were introduced in [13] as a common abstraction for applications in access control, property testing and data structures.

Definition 1.1 (TC-spanner). *Given a directed graph $G = (V, E)$ and an integer $k \geq 1$, a k-**transitive-closure-spanner** (k-**TC-spanner**) of G is a directed graph $H = (V, E_H)$ with the following properties:*

1. *E_H is a subset of the edges in the transitive closure of G.*
2. *For all vertices $u, v \in V$, if $d_G(u, v) < \infty$, then $d_H(u, v) \leq k$.*

Thus, a k-transitive-closure-spanner (or k-TC-spanner) is a graph with small diameter that preserves the connectivity of the original graph. In the applications above, the goal is to find the sparsest k-TC-spanner for a given k and G. The number of edges in the sparsest k-TC-spanner of G is denoted by $S_k(G)$.

Our Contributions. The contributions of this work fall into two categories: (1) We show that a local monotonicity reconstructor, as defined by Saks and Seshadhri [14], for a directed acyclic graph G_n implies a sparse 2-TC-spanner of G_n, providing a new technique for proving lower bounds for local monotonicity reconstructors. (2) We present tight upper and lower bounds on the size of the sparsest 2-TC-spanners of the directed hypercube and hypergrid. These bounds imply tighter lower bounds for local monotonicity reconstructors for these graphs that nearly match the upper bounds given in [14].

1.1 Lower Bounds for Local Monotonicity Reconstruction

Property-preserving data reconstruction was introduced in [15]. In this model, a reconstruction algorithm, called a *filter*, sits between a *client* and a *dataset*. A dataset is viewed as a function $f : \mathcal{D} \to \mathcal{R}$. The client accesses the dataset using *queries* of the form $x \in \mathcal{D}$ to the filter. The filter *looks up* a small number of values in the dataset and outputs $g(x)$, where g must satisfy some fixed *structural* property $\mathcal{P}$. Extending this notion, Saks and Seshadhri [14] defined *local* reconstruction. A filter is *local* if it allows for a local (or distributed) implementation: namely, if the output function g does not depend on the order of the queries.

Definition 1.2 (Local filter). *A* local filter *for reconstructing property $\mathcal{P}$ is an algorithm A that has oracle access to a function $f : \mathcal{D} \to \mathcal{R}$, and to an auxiliary random string ρ (the "random seed"), and takes as input $x \in \mathcal{D}$. For fixed f and ρ, A runs deterministically on input x to produce an output $A_{f,\rho}(x) \in \mathcal{R}$. (Note that a local filter has no internal state to store previously made queries.) The function $g(x) = A_{f,\rho}(x)$ output by the filter must satisfy the following conditions:*

- *For each f and ρ, the function g must satisfy $\mathcal{P}$.*
- *If f satisfies $\mathcal{P}$, then g must be identical to f with probability at least $1-\delta$, for some* error probability $\delta \leq 1/3$. *The probability is taken over ρ.*

In answering query $x \in \mathcal{D}$, the filter A may ask for values of f at domain points of its choice (possibly adaptively) using its oracle access to f. Each such access made to the oracle is called a *lookup* to distinguish it from the client query x. A local filter is *non-adaptive* if the set of domain points that the filter looks up to answer an input query x does not depend on answers given by the oracle.

In [14], the authors also required that g must be sufficiently close to f: *With high probability (over the choice of ρ), $Dist(g,f) \leq B(n) \cdot Dist(f,\mathcal{P})$, where $B(n)$ is called the* error blow-up. ($Dist(g,f)$ is the number of points in the domain on which f and g differ. $Dist(f,\mathcal{P})$ is $\min_{g\in\mathcal{P}} Dist(g,f)$.) If a local filter along with Definition 1.2 satisfies this condition, we call it *distance-respecting*.

Local Monotonicity Reconstructors. The most studied property in the local reconstruction model is monotonicity of functions [14, 15]. A (distance-respecting) filter for monotonicity can be used, for example, when a program runs correctly only if its input is sorted. Then, instead of accessing the input directly, the program can access it via a filter, which ensures that the program always sees a sorted input, with small corrections when necessary. A local filter can be implemented in a distributed manner with an additional guarantee that every program run on the same not-quite-sorted input will see the same corrected version. This can be done by supplying the same random seed to each copy of the filter.

To define monotonicity of functions, consider an n-element poset V_n and let $G_n = (V_n, E)$ be the relation graph, *i.e.,* the Hasse diagram, for V_n. A function $f : V_n \to \mathbb{R}$ is called *monotone* if $f(x) \leq f(y)$ for all $(x,y) \in E$. We particularly focus on posets which have the *directed hypergrid* graph as its relation graph. The *directed hypergrid*, denoted $\mathcal{H}_{m,d}$, has vertex set $\{1,2,\ldots,m\}^d$ and edge set $\{(x,y) : \exists$ unique $i \in \{1,\ldots,d\}$ such that $y_i - x_i = 1$ and for $j \neq i, y_j = x_j\}$. For the special case $m = 2$, $\mathcal{H}_{2,d}$ is called a *hypercube* and is also denoted by $\mathcal{H}_d$. A monotonicity filter needs to ensure that the output function g is monotone. For instance, if G_n is a directed line, $\mathcal{H}_{n,1}$, the filter needs to ensure that the output sequence specified by g is sorted. To motivate reconstructors for hypergrids, consider the following scenario: An admissions office assigns a "score" to rate applicants for admission, where the score consists of d attributes, such as the applicant's GPA, SAT results, essay quality, etc. Based on these scores, some complicated (third-party) algorithm outputs the probability that a given applicant should be accepted. The admissions office wants to make sure "on the fly" that strictly better applicants are given higher probability, that is, probabilities are *monotone* in scores. A hypergrid monotonicity filter may be used here.

In [14], the authors give a *distance-respecting* local monotonicity filter for the directed hypergrid, $\mathcal{H}_{m,d}$, that makes $(\log m)^{O(d)}$ lookups per query. No non-trivial monotonicity filter for the hypercube $\mathcal{H}^d$ (performing $o(2^d)$ lookups per query) is known. One of the monotonicity filters in [15] is a local filter for the directed line $\mathcal{H}_{m,1}$ with $O(\log m)$ lookups per query (but a worse error blow up

than in [14]). As observed in [14], this upper bound is tight. A lower bound of $2^{\alpha d}$, on the number of lookups per query for a *distance-respecting* local monotonicity filter on $\mathcal{H}_d$ with *error blow-up* $2^{\beta d}$, where α, β are sufficiently small constants, appeared in [14]. Notably, all known local filters for the monotonicity property are *non-adaptive*. Here we focus on the non-adaptive case as well.

We show how to construct sparse 2-TC-spanners from local monotonicity reconstructors with low lookup complexity. These constructions, together with our lower bounds on the size of 2-TC-spanners of the hypergrid and hypercube (Section 1.2), imply lower bounds on lookup complexity of local monotonicity reconstructors for these graphs with arbitrary error blow-up. Our transformation is for non-adaptive reconstructors and is stated in Theorem 1.1.

Theorem 1.1 (Transformation from non-adaptive Local Monotonicity Reconstructors to 2-TC-spanners). *Let $G_n = (V_n, E)$ be a poset on n nodes. Suppose there is a* non-adaptive *local monotonicity reconstructor A for G_n that looks up at most $\ell(n)$ values to answer any query $x \in V_n$ and has* error *probability at most δ. Then there is a 2-TC-Spanner of G_n with at most $O(n\ell(n) \cdot \lceil \log n / \log(1/\delta) \rceil)$ edges.*

In Theorem 1.1 when δ is sufficiently small, the bounds on the 2-TC-Spanner size become $O(n\ell(n))$. Theorem 1.1, together with the lower bounds in Theorem 1.3 and Theorem 1.2 (presented in the next section), implies the following lower bounds on the lookup complexity of local monotonicity reconstructors for these graphs with arbitrary error blow-up.

Corollary 1.1. *Consider a nonadaptive local monotonicity filter with constant error probability δ. If the filter is for functions $f : \mathcal{H}_{m,d} \to \mathbb{R}$, it must perform $\Omega\left(\frac{\log^{d-1} m}{d^d (2 \log\log m)^{d-1}}\right)$ lookups per query. If the filter is for functions $f : \mathcal{H}_d \to \mathbb{R}$, it must perform $\Omega\left(2^{\alpha d}/d\right)$ lookups per query, where $\alpha \geq 0.1620$.*

Prior to this work, no lower bounds for monotonicity reconstructors on $\mathcal{H}_{m,d}$ with dependence on both m and d were known. Unlike the bound in [14], our lower bounds hold for any error blow-up and for non-distance-respecting filters. Our bounds are tight for non-adaptive reconstructors. Specifically, for the hypergrid $\mathcal{H}_{m,d}$ of constant dimension d, the number of lookups is $(\log m)^{\Theta(d)}$, and for the hypercube $\mathcal{H}_d$, it is $2^{\Theta(d)}$ for any error blow-up.

Testers vs. Reconstructors. In [13], it was shown how to construct monotonicity testers from 2-TC-spanners. Unlike in the application to monotonicity testing, here we show how to use *lower bounds* on the size of 2-TC-spanners to prove *lower bounds* on complexity of local monotonicity reconstuctors. Lower bounds on the size of 2-TC-spanners do not imply corresponding lower bounds on monotonicity testers. For example, the best monotonicity tester on $\mathcal{H}_d$ runs in $O(d^2)$ time [16, 17], while, as shown in Theorem 1.3, every 2-TC-spanner of $\mathcal{H}_d$ must have size exponential in d.

1.2 Our Results on 2-TC-Spanners of the Hypercube and Hypergrid

Our main theorem gives a set of explicit bounds on $S_2(\mathcal{H}_{m,d})$:

Theorem 1.2 (Hypergrid). *Let $S_2(\mathcal{H}_{m,d})$ denote the number of edges in the sparsest 2-TC-spanner of $\mathcal{H}_{m,d}$. Then*[5] *for $m \geq 3$,*

$$\Omega\left(\frac{m^d \log^d m}{(2d \log\log m)^{d-1}}\right) = S_2(\mathcal{H}_{m,d}) \leq m^d \log^d m.$$

The upper bound in Theorem 1.2 follows from a general construction of k-TC-spanners for graph products for arbitrary $k \geq 2$, presented in the full version. The lower bound is the most technically difficult part of our work. It is proved by a reduction of the 2-TC-spanner construction for $[m]^d$ to that for the $2 \times [m]^{d-1}$ grid and then directly analyzing the number of edges required for a 2-TC-spanner of $2 \times [m]^{d-1}$. We show a tradeoff between the number of edges in the 2-TC-spanner of the $2 \times [m]^{d-1}$ grid that stay within the hyperplanes $\{1\} \times [m]^{d-1}$ and $\{2\} \times [m]^{d-1}$ versus the number of edges that cross from one hyperplane to the other. The proof proceeds in multiple stages. Assuming an upper bound on the number of edges staying within the hyperplanes, each stage is shown to contribute a substantial number of new edges crossing between the hyperplanes. The proof of this tradeoff lemma is already non-trivial for $d = 2$ and is presented in Section 3. The proof for $d > 2$ is deferred to the full version of the paper.

While Theorem 1.2 is most useful when m is large and d is small, in Section 4 we present bounds on $S_2(\mathcal{H}_{m,d})$ which are optimal up to a factor of d^{2m} and, thus, supersede the bounds from Theorem 1.2 when m is small. The general form of these bounds is a somewhat complicated combinatorial expression but they can be estimated numerically. Specifically, $S_2(\mathcal{H}_{m,d}) = 2^{c_m d}\,\mathrm{poly}(d)$, where $c_2 \approx 1.1620$, $c_3 \approx 2.03$, $c_4 \approx 2.82$ and $c_5 \approx 3.24$, each significantly smaller than the exponents corresponding to the transitive closure sizes for the different m.

As a special case of the above, for $m = 2$ we obtain the following theorem for the hypercube. The proof of this theorem is omitted from this version.

Theorem 1.3 (Hypercube). *Let $S_2(\mathcal{H}_d)$ be the number of edges in the sparsest 2-TC-spanner of $\mathcal{H}_d$. Then*

$$\Omega(2^{cd}) = S_2(\mathcal{H}_d) = O(d^3 2^{cd}), \text{ where } c \approx 1.1620.$$

As a comparison point for our bounds, note that the obvious bounds on $S_2(\mathcal{H}_d)$ are the number of edges in the d-dimensional hypercube, $2^{d-1}d$, and the number of edges in the transitive closure of $\mathcal{H}_d$, which is $3^d - 2^d$. (An edge in the transitive closure of $\mathcal{H}_d$ has 3 possibilities for each coordinate: both endpoints are 0, both endpoints are 1, or the first endpoint is 0 and the second is 1. This includes self-loops, so we subtract the number of vertices in $\mathcal{H}_d$ to get the desired quantity.) Thus, $2^{d-1}d \leq S_2(\mathcal{H}_d) \leq 3^d - 2^d$. Similarly, the straightforward bounds on the number of edges in a 2-TC-spanner of $\mathcal{H}_{m,d}$ in terms of the number

[5] Logarithms are always to base 2 unless otherwise indicated.

of edges in the directed grid and in its transitive closure are $dm^{d-1}(m-1)$ and $\left(\frac{m^2+m}{2}\right)^d - m^d$, respectively.

1.3 Previous work on bounding S_k for other families of graphs

Thorup [18] considered a special case of TC-spanners of graphs G that have at most twice as many edges as G, and conjectured that for all directed graphs G on n nodes there are such k-TC-spanners with k polylogarithmic in n. He proved this for planar graphs [19], but Hesse [20] gave a counterexample for general graphs by constructing a family for which all $n^{\frac{1}{17}}$-TC-spanners need $n^{1+\Omega(1)}$ edges. TC-spanners were studied for directed trees: implicitly in [17, 21–24] and explicitly in [25]. For the directed line, [21] (and later, [22]) expressed $S_k(\mathcal{H}_{n,1})$ in terms of the inverse Ackermann function.

Lemma 1.1 ([21, 22, 13]). *Let $S_k(\mathcal{H}_{n,1})$ denote the number of edges in the sparsest k-TC-spanner of the directed line $\mathcal{H}_{n,1}$. Then $S_2(\mathcal{H}_{n,1}) = \Theta(n \log n)$, $S_3(\mathcal{H}_{n,1}) = \Theta(n \log\log n)$, $S_4(\mathcal{H}_{n,1}) = \Theta(n \log^* n)$ and, more generally, $S_k(\mathcal{H}_{n,1}) = \Theta(n\lambda_k(n))$ where $\lambda_k(n)$ is the inverse Ackermann function.*

The same bound holds for directed trees [21, 23, 25]. An $O(n \log n \cdot \lambda_k(n))$ bound on S_k for H-minor-free graph families (e.g., bounded genus and bounded tree-width graphs) was given in [13].

Notation. For a positive integer m, we denote $\{1, \ldots, m\}$ by $[m]$. For $x \in \{0,1\}^d$, we use $|x|$ to denote the weight of x, that is, the number of non-zero coordinates in x. Level i in a hypercube contains all vertices of weight i. The partial order $\preceq$ on the hypergrid $\mathcal{H}_{m,d}$ is defined as follows: $x \preceq y$ for two vertices $x, y \in [m]^d$ iff $x_i \leq y_i$ for all $i \in [d]$. Similarly, $x \prec y$, if x and y are distinct vertices in $[m]^d$ satisfying $x \preceq y$. Vertices x and y are *comparable* if either y is *above* x (that is, $x \preceq y$) or y is *below* x (that is, $y \preceq x$). We denote a path from v_1 to v_ℓ, consisting of edges $(v_1, v_2), (v_2, v_3), \ldots, (v_{\ell-1}, v_\ell)$ by $(v_1, \ldots, v_\ell)$.

2 From Monotonicity Reconstructors to 2-TC-spanners

In this section, we prove Theorem 1.1.

Proof (of Theorem 1.1). Let A be a local reconstructor given by the statement of the theorem. Let $\mathcal{F}$ be the set of pairs (x, y) with x, y in V_n such that $x \prec y$. Then, $\mathcal{F}$ is of size at most $\binom{n}{2}$. Given $(x, y) \in \mathcal{F}$, let $\mathsf{cube}(x, y)$ be the set $\{z \in V_n : x \preceq z \preceq y\}$. Define function $f^{(x,y)}(v)$ to be 1 on all $v \succeq x$ and all $v \succeq y$, and 0 everywhere else. Also, define function $f^{(\overline{x,y})}(v)$, which is identical to $f^{(x,y)}(v)$ for all $v \notin \mathsf{cube}(x, y)$ and 0 for $v \in \mathsf{cube}(x, y)$. Both, $f^{(x,y)}$ and $f^{(\overline{x,y})}$, are monotone functions for all $(x, y) \in \mathcal{F}$. Let A_ρ be the deterministic algorithm which runs A with the random seed fixed to ρ. We say a string ρ is

good for $(x, y) \in \mathcal{F}$ if filter A_ρ on input $f^{(x,y)}$ returns $g = f^{(x,y)}$ *and* on input $f^{(\overline{x,y})}$ returns $g = f^{(\overline{x,y})}$.

Now we show that there exists a set S of size $s \leq \lceil 2 \log n / \log(1/2\delta) \rceil$, consisting of strings used as random seeds by A, such that for every $(x, y) \in \mathcal{F}$ some string $\rho \in S$ is good for (x, y). We choose S by picking strings used as random seeds uniformly and independently at random. Since A has error probability at most δ, we know that for every monotone f, with probability at least $1 - \delta$ (with respect to the choice of ρ), the function $A_{f,\rho}$ is identical to f. Then, for fixed $(x, y) \in \mathcal{F}$ and uniformly random ρ,

$$\begin{aligned}\Pr[\rho \text{ is not } \textit{good} \text{ for } (x,y)] \leq \quad & \Pr[A_\rho \text{ on input } f^{(x,y)} \text{ fails to output } f^{(x,y)}] \\ & + \Pr[A_\rho \text{ on input } f^{(\overline{x,y})} \text{ fails to output } f^{(\overline{x,y})}] \leq 2\delta.\end{aligned}$$

Since strings in S are chosen independently, $\Pr[\text{no } \rho \in S \text{ is good for } (x, y)] \leq (2 \cdot \delta)^s$, which, for $s = \lceil 2 \log n / \log(1/2\delta) \rceil$, is at most $1/n^2 < 1/|\mathcal{F}|$. By a union bound over $\mathcal{F}$, $\Pr[\text{for some } (x, y) \in \mathcal{F}, \text{ no } \rho \in S \text{ is good for } (x, y)] < 1$. Thus, there exists a set S with required properties.

We construct our 2-TC-spanner $H = (V_n, E_H)$ of G_n using set S described above. Let $\mathcal{N}_\rho(x)$ be the set consisting of x and all vertices looked up by A_ρ on query x. (Note that the set $\mathcal{N}_\rho(x)$ is well-defined since algorithm A is assumed to be *non-adaptive*). For each string $\rho \in S$ and each vertex $x \in V_n$, connect x to all comparable vertices in $\mathcal{N}_\rho(x)$ (other than itself) and orient these edges according to their direction in G_n.

We prove H is a 2-TC-Spanner as follows. Suppose not, *i.e.,* there exists $(x, y) \in \mathcal{F}$ with no path of length at most 2 in H from x to y. Consider $\rho \in S$ which is *good* for (x, y). Define function h by setting $h(v) = f^{(x,y)}(v)$ for all $v \notin \mathsf{cube}(x, y)$. Then $h(v) = f^{(\overline{x,y})}(v)$ for all $v \notin \mathsf{cube}(x, y)$, by definition of $f^{(\overline{x,y})}$. For a $v \in \mathsf{cube}(x, y)$, set $h(v)$ to 1 for $v \in \mathcal{N}_\rho(x)$ and to 0 for $v \in \mathcal{N}_\rho(y)$. All unassigned points are set to 0. By the assumption above, $\mathcal{N}_\rho(x) \cap \mathcal{N}_\rho(y)$ does not contain any points in $\mathsf{cube}(x, y)$. Therefore, h is well-defined. Since ρ is *good* for (x, y) and h is identical to $f^{(x,y)}$ for all lookups made on query x, $A_\rho(x) = h(x) = 1$. Similarly, $A_\rho(y) = h(y) = 0$. But $x \prec y$, so $A_{h,\rho}(v)$ is not monotone. Contradiction.

The number of edges in H is at most

$$\sum_{x \in V_n, \rho \in S} |\mathcal{N}_\rho(x)| \leq n \cdot \ell(n) \cdot s \leq n\ell(n) \cdot \lceil 2 \log n / \log(1/2\delta) \rceil. \qquad \square$$

3 2-TC-Spanners for low-dimensional hypergrids

In this section, we describe the proof of Theorem 1.2 which gives explicit bounds on the size of the sparsest 2-TC-spanner for $\mathcal{H}_{m,d}$. The upper bound in Theorem 1.2 follows straightforwardly from a more general statement about TC-spanners of product graphs; details are in the full version. Here, we show the lower bound on $S_2(\mathcal{H}_{m,d})$. Actually, in this extended abstract, we treat only the special case of this lower bound for $d = 2$, since it already contains most of the

difficulty of the larger dimensional case. The extension to arbitrary dimension is deferred to the full version due to space constraints.

Theorem 3.1. *Any 2-TC-spanner of the 2-dimensional grid $\mathcal{H}_{m,2}$ must have $\Omega\left(\frac{m^2 \log^2 m}{\log\log m}\right)$ edges.*

One way to prove the $\Omega(m \log m)$ lower bound on the size of a 2-TC-spanner for the directed line $\mathcal{H}_{m,1}$, stated in Lemma 1.1, is to observe that at least $\lfloor \frac{m}{2} \rfloor$ edges are cut when the line is halved: namely, at least one per vertex pair $(v, m - v + 1)$ for all $v \in \left[\lfloor \frac{m}{2} \rfloor\right]$. Continuing to halve the line recursively, we obtain the desired bound.

A natural extension of this approach to proving a lower for the grid is to recursively halve the grid along both dimensions, hoping that each such operation on an $m \times m$ grid cuts $\Omega(m^2 \log m)$ edges. This would imply that the size $S(m)$ of a 2-TC-spanner of the $m \times m$ grid satisfies the recurrence $S(m) = 4S(m/2) + \Omega(m^2 \log m)$; that is, $S(m) = \Omega(m^2 \log^2 m)$, matching the upper bound in Theorem 1.2.

An immediate problem with this approach is that in some 2-TC-spanners of the grid only $O(m^2)$ edges connect vertices in different quarters. One example of such a 2-TC-spanner is the graph containing the transitive closure of each quarter and only at most $3m^2$ edges crossing from one quarter to another: namely, for each node u and each quarter q with vertices comparable to u, this graph contains an edge (u, v_q), where v_q is the smallest node in q comparable to u.

The TC-spanner in the example above is not optimal because it has too many edges inside the quarters. The first step in our proof of Theorem 3.1 is understanding the tradeoff between the number of edges *crossing* the cut and the number of edges *internal* to the subgrids, resulting from halving the grid along some dimension. The simplest manifestation of this tradeoff occurs when a $2 \times m$ grid is halved into two lines. (In the case of one line, there is no trade off: the $\Omega(m)$ bound on the number of crossing edges holds even if each half-line contains all edges of its transitive closure.) Lemma 3.1 formulates the tradeoff for the two-line case, while taking into account only edges needed to connect comparable vertices on different lines by paths of length at most 2:

Lemma 3.1 (Two-Lines Lemma). *Let U be a graph with vertex set $[2] \times [m]$ that contains a path of length at most 2 from u to v for every $u \in \{1\} \times [m]$ and $v \in \{2\} \times [m]$, where $u \preceq v$. An edge (u, v) in U is called* internal *if $u_1 = v_1$, and* crossing *otherwise. If U contains at most $\frac{m \log^2 m}{32}$ internal edges, it must contain at least $\frac{m \log m}{16 \log\log m}$ crossing edges.*

Note that if the number of internal edges is unrestricted, a 2-TC-spanner of $\mathcal{H}_{m,2}$ may have only m crossing edges.

Proof. The proof proceeds in $\frac{\log m}{2 \log\log m}$ stages dealing with pairwise disjoint sets of crossing edges. In each stage, we show that U contains at least $\frac{m}{8}$ crossing edges in the prescribed set.

In the first stage, divide U into $\log^2 m$ blocks, each of length $\frac{m}{\log^2 m}$: namely, a node (v_1, v_2) is in block i if $v_2 \in \left[\frac{(i-1)\cdot m}{\log^2 m} + 1, \frac{i \cdot m}{\log^2 m}\right]$. Call an edge *long* if it starts and ends in different blocks, and *short* otherwise. Assume, for contradiction, that U contains fewer than $\frac{m}{8}$ long crossing edges.

Call a node (v_1, v_2) *low* if $v_1 = 1$ (*high* if $v_1 = 2$), and *left* if $v_2 \in \left[\frac{m}{2}\right]$ (*right* otherwise). Also, call an edge (u, v) *low-internal* if $u_1 = v_1 = 1$ and *high-internal* if $u_1 = v_1 = 2$. Let L be the set of low left nodes that are not incident to long crossing edges. Similarly, let R be the set of high right nodes that are not incident to long crossing edges. Since there are fewer than $\frac{m}{8}$ long crossing edges, $|L| > \frac{m}{4}$ and $|R| > \frac{m}{4}$.

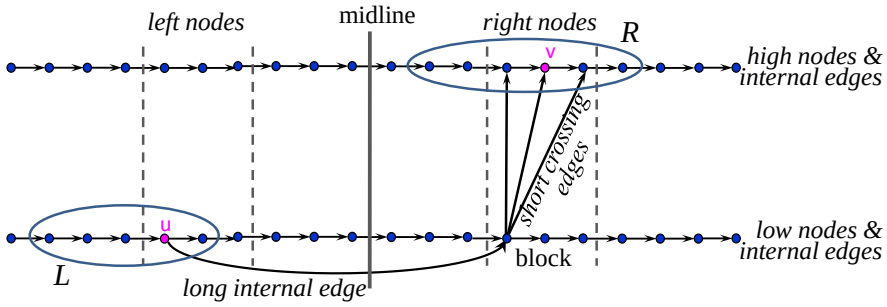

Fig. 1. Illustration of the first stage in the proof of Lemma 3.1.

A node $u \in L$ can connect to a node $v \in R$ via a path of length at most 2 only by using a long internal edge. Observe that each long low-internal edge can be used by at most $\frac{m}{\log^2 m}$ such pairs (u, v): one low node u and high nodes v from one block. This is illustrated in Figure 1. Analogously, every long high-internal edge can be used by at most $\frac{m}{\log^2 m}$ such pairs. Since $|L| \cdot |R| > \frac{m^2}{16}$ pairs in $L \times R$ connect via paths of length at most 2, graph U contains more than $\frac{m^2}{16} \cdot \frac{\log^2 m}{m} = \frac{m \log^2 m}{16}$ long internal edges, which is a contradiction.

In each subsequent stage, call blocks used in the previous stage *megablocks*, and denote their length by B. Subdivide each megablock into $\log^2 m$ blocks of equal size. Call an edge *long* if it starts and ends in different blocks, but stays within one megablock. Assume, for contradiction, that U contains fewer than $\frac{m}{8}$ long crossing edges.

Call a node (v_1, v_2) *left* if it is in the left half of its megablock, that is, if $v_2 \leq \frac{\ell + r}{2}$ whenever (v_1, v_2) is in a megablock $[2] \times \{\ell, \ldots, r\}$. (Call it *right* otherwise). Consider megablocks containing fewer than $\frac{B}{4}$ long crossing edges each. By an averaging argument, at least $\frac{m}{2B}$ megablocks are of this type. (Recall that there are $\frac{m}{B}$ megablocks in total). Within each such megablock more than $\frac{B}{4}$ low left nodes and more than $\frac{B}{4}$ high right nodes have no incident long crossing

edges. By the argument from the first stage, each such megablock contributes more than $\frac{B^2}{16b}$ long internal edges, where $b = \frac{B}{\log^2 m}$ is the size of the blocks. Hence there must be more than $\frac{B^2}{16b} \cdot \frac{m}{2B} = \frac{m \log^2 m}{32}$ long internal edges, which is a contradiction to the fact that U contains at most $\frac{m \log^2 m}{32}$ internal edges.

We proceed to the next stage until each block is of length 1. Therefore, the number of stages, t, satisfies $\frac{m}{\log^{2t} m} = 1$. That is, $t = \frac{\log m}{2 \log \log m}$, and each stage contributes $\frac{m}{8}$ new crossing edges, as desired. □

Next we generalize Lemma 3.1 to understand the tradeoff between the number of internal edges and crossing edges resulting from halving a 2-TC-spanner of a $2\ell \times m$ grid with the usual partial order.

Lemma 3.2. *Let S be a 2-TC-spanner of the directed $[2\ell] \times [m]$ grid. An edge (u, v) in S is called* internal *if $u_1, v_1 \in [\ell]$ or $u_1, v_1 \in \{\ell + 1, \ldots, 2\ell\}$, and* crossing *otherwise. If S contains at most $\frac{\ell m \log^2 m}{64}$ internal edges, it must contain at least $\frac{\ell m \log m}{32 \log \log m}$ crossing edges.*

Proof. For each $i \in [\ell]$, we match the lines $\{i\} \times [m]$ and $\{2\ell - i + 1\} \times [m]$. Observe that a path of length at most 2 between the matched lines cannot use any edges with both endpoints in $\{i+1, \ldots, 2\ell - i\} \times [m]$. We modify S to ensure that there are no edges with only one endpoint in $\{i+1, \ldots, 2\ell - i\} \times [m]$ for all $i \in [\ell]$, and then apply Lemma 3.1 to the matched pairs of lines.

Call the $[\ell] \times [m]$ subgrid and all vertices and edges it contains *low*, and the remaining $\{\ell+1, \ldots, 2\ell\} \times [m]$ subgrid and its vertices and edges *high*. Transform S into S' as follows: change each low internal edge (u, v) to $(u, (u_1, v_2))$, change each high internal edge (u, v) to $((v_1, u_2), v)$, and finally change each crossing edge $((i_1, j_1), (2\ell - i_2 + 1, j_2))$ to $((i, j_1), (2\ell - i + 1, j_2))$, where $i = \min(i_1, i_2)$. Intuitively, we are projecting the edges in S to be fully contained in one of the matched pairs of lines, while preserving whether the edge is internal or crossing. Crossing edges are projected onto the outer matched pair of lines chosen from the two pairs that contain the endpoints of a given edge.

Clearly, S' contains at most the number of internal (crossing) edges as S. Observe that S' contains a path of length at most 2 from u to v for every comparable pair (u, v) where u is low, v is high, and u and v belong to the same pair of matched lines. Indeed, since S is a 2-TC-spanner, it contains either the edge (u, v) or a path (u, w, v). In the first case, S' also contains (u, v). In the second case, if (u, w) is a crossing edge S' contains $(u, (v_1, w_2), v)$, and if (u, w) is an internal edge S' contains $(u, (u_1, w_2), v)$. As claimed, each edge in S' belongs to one of the matched pairs of lines.

Finally, we apply Lemma 3.1. If S contains at most $\frac{\ell m \log^2 m}{64}$ internal edges, then so does S', and so at least half $\left(\text{i.e., } \frac{\ell}{2}\right)$ of the matched line pairs each contain at most $\frac{m \log^2 m}{32}$ internal edges. By Lemma 3.1, each of these pairs contributes at least $\frac{m \log m}{16 \log \log m}$ crossing edges. Thus S' must contain at least $\frac{\ell m \log m}{32 \log \log m}$ crossing edges. Since S contains as many crossing edges as S', the lemma follows. □

Now we prove Theorem 3.1 by recursively halving $\mathcal{H}_{m,2}$ along the horizontal dimension. Some resulting $\ell \times m$ subgrids may violate Lemma 3.2, but we can guarantee that the lemma holds for a constant fraction of the recursive steps for which $\ell \geq \sqrt{m}$. This is sufficient for obtaining the lower bound in the theorem.

Proof (of Theorem 3.1). Assume m is a power of 2 for simplicity. For each step $i \in \{1, \ldots, \frac{1}{2}\log m\}$, partition $\mathcal{H}_{m,2}$ into the following 2^{i-1} equal-sized subgrids: $\{1, \ldots, l_i\} \times [m]$, $\{l_i + 1, \ldots, 2l_i\} \times [m]$, $\ldots$, $\{m - l_i + 1, \ldots, m\} \times [m]$ where $l_i = m/2^{i-1}$. For each of these subgrids, define internal and crossing edges as in Lemma 3.2. Now, suppose that there exists a step i such that at least half of the 2^{i-1} subgrids have $> \frac{l_i m \log^2 m}{64}$ internal edges. Since at a fixed i, the subgrids are disjoint, there are $2^{i-1}\Omega(l_i m \log^2 m) = \Omega(m^2 \log^2 m)$ edges in S, proving the theorem. On the other hand, suppose that for every $i \in \{1, \ldots, \frac{1}{2}\log m\}$, at least half of the 2^{i-1} subgrids have $\leq \frac{l_i m \log^2 m}{64}$ internal edges. Then, applying Lemma 3.2, the number of crossing edges in those subgrids is $\geq \frac{l_i m \log m}{32 \log\log m}$. Counting over all steps i and for all appropriate subgrids from those steps, the number of edges in S is bounded by $\Omega\left(m^2 \log m \frac{\log m}{\log\log m}\right) = \Omega\left(m^2 \frac{\log^2 m}{\log\log m}\right)$. □

In the full version, we extend the above proof to establish lower bounds on $S_2(\mathcal{H}_{m,d})$ for arbitrary $d \geq 2$. The main technical deferred result is a trade-off lemma between internal and crossing edges with respect to two $(d-1)$-dimensional hyperplanes. An important part of the generalization is the appropriate definition of the notions of blocks and megablocks, so that the iterative argument in the proof of Lemma 3.1 applies in the high-dimensional setting.

4 2-TC-spanners for high-dimensional hypergrids

Theorem 4.1 gives matching upper and lower bounds up to a d^{2m} factor in terms of an expression involving binomial coefficients. This result supersedes the results of the previous section when, for instance, m is constant and d is growing.

Before stating Theorem 4.1, we introduce some notation.

Definition 4.1. *For the hypergrid $\mathcal{H}_{m,d}$, define a* level *to be a set of vertices, indexed by vector $\mathbf{i} \in [d]^m$ with $i_1 + \cdots + i_m = d$, that consists of vertices $x = (x_1, \ldots, x_d) \in [m]^d$ containing i_1 positions of value 1, i_2 positions of value 2, $\ldots$, and i_m positions of value m.*

Notice that the number of vertices in level $\mathbf{i} = (i_1, i_2, \ldots, i_m)$ is the multinomial coefficient

$$\binom{d}{\mathbf{i}} = \binom{d}{i_1, ..., i_d} = \binom{d}{i_1}\binom{d-i_1}{i_2}\binom{d-i_1-i_2}{i_3}\cdots\binom{d-\sum_{l=1}^{m-1} i_l}{i_m}.$$

Indeed, there are $\binom{d}{i_1}$ choices for the coordinates of value 1. For each such choice there are $\binom{d-i_1}{i_2}$ choices for the coordinates of value 2, and repeating this argument one obtains the above expression.

For levels $\mathbf{i}, \mathbf{j} \in [d]^m$, say $\mathbf{j}$ *majorizes* $\mathbf{i}$, denoted $\mathbf{j} \succ \mathbf{i}$, if $\mathbf{j}$ contains a vertex which is above some vertex in $\mathbf{i}$, that is, if $\sum_{\ell=t}^{m} j_\ell \geq \sum_{\ell=t}^{m} i_\ell$ for all $t \in \{m, m-1, ..., 1\}$.

For $\mathbf{j} \succ \mathbf{i}$, the number of vertices y at level $\mathbf{i}$ comparable to a fixed vertex x at level $\mathbf{j}$ is $\mathcal{M}(\mathbf{i}, \mathbf{j})$:

$$\binom{j_m}{i_m}\binom{j_m + j_{m-1} - i_m}{i_{m-1}}\binom{j_m + j_{m-1} + j_{m-2} - i_m - i_{m-1}}{i_{m-2}} \dots \binom{\sum_{l=1}^{m} j_l - \sum_{l=2}^{m} i_l}{i_1}.$$

Indeed, there are $\binom{j_m}{i_m}$ choices for the coordinates of value m in y. For each such choice, there are $\binom{j_m + j_{m-1} - i_m}{i_{m-1}}$ choices for the coordinates of value $m-1$ in y, and one can repeat this argument to obtain the claimed expression.

For $\mathbf{j} \succ \mathbf{i}$, the number of vertices y at level $\mathbf{j}$ comparable to a fixed vertex x at level $\mathbf{i}$ is

$$\mathcal{N}(\mathbf{i}, \mathbf{j}) = \frac{\mathcal{M}(\mathbf{i}, \mathbf{j})\binom{d}{\mathbf{j}}}{\binom{d}{\mathbf{i}}}.$$

Indeed, there are $\mathcal{M}(\mathbf{i}, \mathbf{j})\binom{d}{\mathbf{j}}$ comparable pairs of vertices in levels $\mathbf{i}$ and $\mathbf{j}$, and level $\mathbf{i}$ contains $\binom{d}{\mathbf{i}}$ vertices. Since, by symmetry, each vertex in $\mathbf{i}$ is comparable to the same number of vertices in level $\mathbf{j}$, we get the desired expression.

Theorem 4.1. *Let*

$$\mathcal{B}(m, d) = \max_{\mathbf{i},\mathbf{j}:\mathbf{j}\succ\mathbf{i}} \min_{\mathbf{k}:\mathbf{i}\prec\mathbf{k}\prec\mathbf{j}} \frac{\mathcal{M}(\mathbf{i}, \mathbf{j})\binom{d}{\mathbf{j}}}{\mathcal{M}(\mathbf{i}, \mathbf{k})\mathcal{N}(\mathbf{k}, \mathbf{j})} \max\left\{\mathcal{M}(\mathbf{i}, \mathbf{k}), \mathcal{N}(\mathbf{k}, \mathbf{j})\right\}.$$

Then the number of edges in the sparsest 2-TC-spanner of the directed hypergrid $\mathcal{H}_{m,d}$ is $O\left(d^{2m}\mathcal{B}(m, d)\right)$ and $\Omega\left(\mathcal{B}(m, d)\right)$.

The proof for the upper bound part of Theorem 4.1 appears in the full version. We now prove the lower bound.

Lemma 4.1. *Any 2-TC-spanner of $\mathcal{H}_{m,d}$ has at least $\Omega(\mathcal{B}(m, d))$ many edges, where $\mathcal{B}(m, d)$ is defined as in Theorem 4.1.*

Proof. Let S be a 2-TC-spanner for $\mathcal{H}_{m,d}$. We will count the edges in S that occur on paths connecting two particular levels of $\mathcal{H}_{m,d}$. Let $P_{\mathbf{i},\mathbf{j}} = \{(v_1, v_2) : v_1 \in \mathbf{i}, v_2 \in \mathbf{j}, v_1 \prec v_2\}$. We will lower bound $e^*_{\mathbf{i},\mathbf{j}}$, the number of edges in the paths of length at most 2 in S, that connect the pairs $P_{\mathbf{i},\mathbf{j}}$. Notice that $|P(\mathbf{i}, \mathbf{j})| = \binom{d}{\mathbf{j}}\mathcal{M}(\mathbf{i}, \mathbf{j})$.

Let $e_{\mathbf{k},\boldsymbol{\ell}}$ denote the number of edges in S that connect vertices in level $\mathbf{k}$ to vertices in level $\boldsymbol{\ell}$. Then

$$e^*_{\mathbf{i},\mathbf{j}} = e_{\mathbf{i},\mathbf{j}} + \sum_{\mathbf{i}\prec\mathbf{k}\prec\mathbf{j}} (e_{\mathbf{i},\mathbf{k}} + e_{\mathbf{k},\mathbf{j}}). \tag{1}$$

We say that a vertex v *covers* a pair of vertices (v_1, v_2) if S contains the edges (v_1, v) and (v, v_2) or, for the special case $v = v_1$, if S contains (v_1, v_2). Let $V^{(\mathbf{k})}_{\mathbf{i},\mathbf{j}}$ be the set of vertices in level $\mathbf{k}$ that cover pairs in $P_{\mathbf{i},\mathbf{j}}$. Let $\alpha_{\mathbf{k}}$ be the fraction of pairs in $P_{\mathbf{i},\mathbf{j}}$ that are covered by the vertices in $V^{(\mathbf{k})}_{\mathbf{i},\mathbf{j}}$. Since each pair in $P_{\mathbf{i},\mathbf{j}}$ must be covered by a vertex in levels $\mathbf{k}$ with $\mathbf{i} \prec \mathbf{k} \prec \mathbf{j}$, we must have $\sum_{\mathbf{i}\prec\mathbf{k}\prec\mathbf{j}} \alpha_{\mathbf{k}} \geq 1$.

For any vertex $v \in V^{(\mathbf{k})}_{\mathbf{i},\mathbf{j}}$, let in_v be the number of incoming edges from vertices of level $\mathbf{i}$ incident to v and let out_v be the number of outgoing edges to vertices of level $\mathbf{j}$ incident to v. For each level $\mathbf{k}$ with $\mathbf{i} \prec \mathbf{k} \prec \mathbf{j}$, since each vertex $v \in V^{(\mathbf{k})}_{\mathbf{i},\mathbf{j}}$ covers $in_v \cdot out_v$ pairs,

$$\sum_{v \in V^{(\mathbf{k})}_{\mathbf{i},\mathbf{j}}} in_v \cdot out_v \geq \alpha_{\mathbf{k}} |P_{\mathbf{i},\mathbf{j}}| \geq \alpha_{\mathbf{k}} \mathcal{M}(\mathbf{i},\mathbf{j}) \binom{d}{\mathbf{j}}. \tag{2}$$

We upper bound $\sum_{v \in V^{(\mathbf{k})}_{\mathbf{i},\mathbf{j}}} in_v \cdot out_v$ as a function of $e_{\mathbf{i},\mathbf{k}} + e_{\mathbf{k},\mathbf{j}}$, and then use Equation (2) to lower bound $e_{\mathbf{i},\mathbf{k}} + e_{\mathbf{k},\mathbf{j}}$. For all $\mathbf{k}$ with $\mathbf{i} \prec \mathbf{k} \prec \mathbf{j}$, variables in_v and out_v satisfy the following constraints:

$$\sum_{v \in V^{(\mathbf{k})}_{\mathbf{i},\mathbf{j}}} in_v \leq e_{\mathbf{i},\mathbf{k}} \leq e_{\mathbf{i},\mathbf{k}} + e_{\mathbf{k},\mathbf{j}}, \qquad \sum_{v \in V^{(\mathbf{k})}_{\mathbf{i},\mathbf{j}}} out_v \leq e_{\mathbf{k},\mathbf{j}} \leq e_{\mathbf{i},\mathbf{k}} + e_{\mathbf{k},\mathbf{j}},$$

$$in_v \leq \mathcal{M}(\mathbf{i},\mathbf{k}) \ \forall v \in V^{(\mathbf{k})}_{\mathbf{i},\mathbf{j}}, \quad out_v \leq \mathcal{N}(\mathbf{k},\mathbf{j}) \ \forall v \in V^{(\mathbf{k})}_{\mathbf{i},\mathbf{j}}.$$

The last two constraints hold because in_v and out_v count the number of edges to a vertex of level $\mathbf{k}$ from vertices of level $\mathbf{i}$, and from a vertex of level $\mathbf{k}$ to vertices of level $\mathbf{j}$, respectively. Using these bounds we obtain

$$\sum_{v \in V^{(\mathbf{k})}_{\mathbf{i},\mathbf{j}}} in_v \cdot out_v \leq \sum_{v \in V^{(\mathbf{k})}_{\mathbf{i},\mathbf{j}}} \mathcal{M}(\mathbf{i},\mathbf{k}) \cdot out_v = \mathcal{M}(\mathbf{i},\mathbf{k}) \cdot \sum_{v \in V^{(\mathbf{k})}_{\mathbf{i},\mathbf{j}}} out_v \leq \mathcal{M}(\mathbf{i},\mathbf{k}) \cdot (e_{\mathbf{i},\mathbf{k}} + e_{\mathbf{k},\mathbf{j}}).$$

Similarly, $\sum_{v \in V^{(\mathbf{k})}_{\mathbf{i},\mathbf{j}}} in_v \cdot out_v \leq \mathcal{N}(\mathbf{k},\mathbf{j}) \cdot (e_{\mathbf{i},\mathbf{k}} + e_{\mathbf{k},\mathbf{j}})$. Therefore,

$$\sum_{v \in V^{(\mathbf{k})}_{\mathbf{i},\mathbf{j}}} in_v \cdot out_v \leq (e_{\mathbf{i},\mathbf{k}} + e_{\mathbf{k},\mathbf{j}}) \min\left\{\mathcal{M}(\mathbf{i},\mathbf{k}), \mathcal{N}(\mathbf{k},\mathbf{j})\right\}.$$

From Equation (2), $e_{\mathbf{i},\mathbf{k}} + e_{\mathbf{k},\mathbf{j}} \geq \alpha_{\mathbf{k}} \mathcal{M}(\mathbf{i},\mathbf{j}) \binom{d}{\mathbf{j}} \frac{1}{\min\{\mathcal{M}(\mathbf{i},\mathbf{k}), \mathcal{N}(\mathbf{k},\mathbf{j})\}}$ for all $\mathbf{i} \prec \mathbf{k} \prec \mathbf{j}$. Applying Equation (1) and the fact that $\sum_{\mathbf{i}\prec\mathbf{k}\prec\mathbf{j}} \alpha_{\mathbf{k}} \geq 1$, we get

$$\begin{aligned} e^*_{\mathbf{i},\mathbf{j}} &= e_{\mathbf{i},\mathbf{j}} + \sum_{\mathbf{i}\prec\mathbf{k}\prec\mathbf{j}} (e_{\mathbf{i},\mathbf{k}} + e_{\mathbf{k},\mathbf{j}}) \geq \sum_{\mathbf{k}} \alpha_{\mathbf{k}} \frac{1}{\min\{\mathcal{M}(\mathbf{i},\mathbf{k}), \mathcal{N}(\mathbf{k},\mathbf{j})\}} \mathcal{M}(\mathbf{i},\mathbf{j}) \binom{d}{\mathbf{j}} \\ &\geq \min_{\mathbf{k}} \frac{1}{\min\{\mathcal{M}(\mathbf{i},\mathbf{k}), \mathcal{N}(\mathbf{k},\mathbf{j})\}} \mathcal{M}(\mathbf{i},\mathbf{j}) \binom{d}{\mathbf{j}} \\ &= \min_{\mathbf{k}} \frac{1}{\mathcal{M}(\mathbf{i},\mathbf{k}) \mathcal{N}(\mathbf{k},\mathbf{j})} \mathcal{M}(\mathbf{i},\mathbf{j}) \binom{d}{\mathbf{j}} \max\left\{\mathcal{M}(\mathbf{i},\mathbf{k}), \mathcal{N}(\mathbf{k},\mathbf{j})\right\}. \end{aligned}$$

Since this holds for arbitrary $\mathbf{i}$ and $\mathbf{j}$, the size of the 2-TC-spanner is $|S| \geq \mathcal{B}(m,d)$. □

References

1. Peleg, D., Schäffer, A.A.: Graph spanners. Journal of Graph Theory **13** (1989) 99–116
2. Cowen, L.: Compact routing with minimum stretch. J. Algorithms **38** (2001) 170–183
3. Cowen, L., Wagner, C.G.: Compact roundtrip routing in directed networks. J. Algorithms **50** (2004) 79–95
4. Peleg, D., Upfal, E.: A trade-off between space and efficiency for routing tables. JACM **36** (1989) 510–530
5. Roditty, L., Thorup, M., Zwick, U.: Roundtrip spanners and roundtrip routing in directed graphs. In: SODA. (2002) 844–851
6. Thorup, M., Zwick, U.: Compact routing schemes. In: ACM Symposium on Parallel Algorithms and Architectures. (2001) 1–10
7. Peleg, D., Ullman, J.D.: An optimal synchronizer for the hypercube. SIAM J. Comput. **18** (1989) 740–747
8. Cohen, E.: Fast algorithms for constructing t-spanners and paths with stretch t. SIAM J. Comput. **28** (1998) 210–236
9. Cohen, E.: Polylog-time and near-linear work approximation scheme for undirected shortest paths. JACM **47** (2000) 132–166
10. Elkin, M.: Computing almost shortest paths. In: PODC. (2001) 53–62
11. Baswana, S., Sen, S.: Approximate distance oracles for unweighted graphs in expected $\tilde{O}(n^2)$ time. ACM Transactions on Algorithms **2** (2006) 557–577
12. Thorup, M., Zwick, U.: Approximate distance oracles. JACM **52** (2005) 1–24
13. Bhattacharyya, A., Grigorescu, E., Jung, K., Raskhodnikova, S., Woodruff, D.P.: Transitive-closure spanners. In: SODA. (2009) 932–941
14. Saks, M., Seshadhri, C.: Local monotonicity reconstruction. SIAM Journal on Computing **39** (2010) 2897–2926
15. Ailon, N., Chazelle, B., Comandur, S., Liu, D.: Property-preserving data reconstruction. Algorithmica **51** (2008) 160–182
16. Goldreich, O., Goldwasser, S., Lehman, E., Ron, D., Samorodnitsky, A.: Testing monotonicity. Combinatorica **20** (2000) 301–337
17. Dodis, Y., Goldreich, O., Lehman, E., Raskhodnikova, S., Ron, D., Samorodnitsky, A.: Improved testing algorithms for monotonicity. In: RANDOM. (1999) 97–108
18. Thorup, M.: On shortcutting digraphs. In: WG. (1992) 205–211
19. Thorup, M.: Shortcutting planar digraphs. Combinatorics, Probability & Computing **4** (1995) 287–315
20. Hesse, W.: Directed graphs requiring large numbers of shortcuts. In: SODA. (2003) 665–669
21. Alon, N., Schieber, B.: Optimal preprocessing for answering on-line product queries. Technical Report 71/87, Tel-Aviv University (1987)
22. Atallah, M.J., Frikken, K.B., Fazio, N., Blanton, M.: Dynamic and efficient key management for access hierarchies. In: ACM Conference on Computer and Communications Security. (2005) 190–202
23. Chazelle, B.: Computing on a free tree via complexity-preserving mappings. Algorithmica **2** (1987) 337–361
24. Yao, A.C.C.: Space-time tradeoff for answering range queries (extended abstract). In: STOC. (1982) 128–136
25. Thorup, M.: Parallel shortcutting of rooted trees. J. Algorithms **23** (1997) 139–159