Laboratoire Bordelais de Recherche en Informatique
UMR 5800 - Université Bordeaux 1, 351, cours de la Libération,
33405 Talence CEDEX, France

Research Report RR-1441-08

On the Locality of Distributed Sparse Spanner Construction

by B. Derbel, C. Gavoille, D. Peleg, and L. Viennot

February, 2008

On the Locality of Distributed Sparse Spanner Construction

Bilel Derbel[*] Cyril Gavoille[†] David Peleg[‡] Laurent Viennot[§]

Abstract

The paper presents a deterministic distributed algorithm that, given $k \geqslant 1$, constructs in k rounds a $(2k-1,0)$-spanner of $O(kn^{1+1/k})$ edges for every n-node unweighted graph. (If n is not available to the nodes, then our algorithm executes in $3k-2$ rounds, and still returns a $(2k-1,0)$-spanner with $O(kn^{1+1/k})$ edges.) Previous distributed solutions achieving such optimal stretch-size trade-off either make use of randomization and provide no performance guarantees, or perform in $\log^{\Omega(1)} n$ rounds, and all require a priori knowledge of n. Based on this algorithm, we propose a second deterministic distributed algorithm that, for every $\epsilon > 0$, constructs a $(1+\epsilon, 2)$-spanner of $O(\epsilon^{-2}n^{3/2})$ edges in $O(\epsilon^{-1})$ rounds, without any prior knowledge on the graph.

Our algorithms are complemented with lower bounds, which hold even under the assumption that n is known to the nodes. It is shown that any (randomized) distributed algorithm requires k rounds in expectation to compute a $(2k-1,0)$-spanner of $o(n^{1+1/(k-1)})$ edges for $k \in \{2,3,5\}$. It is also shown that for every $k > 1$, any (randomized) distributed algorithm that constructs a spanner with fewer than $n^{1+1/k+\epsilon}$ edges in at most n^{ϵ} expected rounds must stretch some distances by an additive factor of $n^{\Omega(\epsilon)}$. In other words, while additive stretched spanners with $O(n^{1+1/k})$ edges may exist, e.g., for $k = 2, 3$, they cannot be computed distributively in a polynomial number of rounds in expectation.

Keywords: distributed algorithms, graph spanners, time complexity.

1 Introduction

1.1 Background

This paper concerns fast deterministic distributed construction of graph spanners. Informally speaking, a graph spanner is a skeleton structure that allows us to represent the underlying network using few edges, in such a way that for any two nodes of the network, the distance in the spanner is stretched by only a small factor. The quality of a spanner is often

[*]Laboratoire d'Informatique Fondamentale de Lille (LIFL), Université des Sciences et Technologies de Lille, France. Bilel.Derbel@lifl.fr.

[†]Laboratoire Bordelais de Recherche en Informatique (LaBRI), Université de Bordeaux, France. gavoille@labri.fr. Supported by the ANR-project "ALADDIN", and the équipe-projet INRIA "CÉPAGE".

[‡]Department of Computer Science and Applied Mathematics, The Weizmann Institute, Rehovot, Israel. david.peleg@weizmann.ac.il. Supported in part by grants from the Israel Science Foundation and the Minerva Foundation.

[§]INRIA, University Paris 7, France. Laurent.Viennot@inria.fr. Supported by the ANR-project "ALADDIN", and the équipe-projet INRIA "GANG".

given by the trade-off between the number of edges it uses and its multiplicative/additive stretch factor, measuring distance preservation. More formally, given a graph G, a subgraph H of G with $V(H) = V(G)$ is an (α, β)-spanner if for every pair of nodes u and v in G, $d_H(u, v) \leqslant \alpha \cdot d_G(u, v) + \beta$. The *stretch* of the spanner is defined as (α, β) and its size as $|E(H)|$. Our interest in graph spanners stems from the fact that spanners are used explicitly or implicitly as key ingredients of various distributed applications, e.g., synchronizers [Awe85, PU89a], compact routing [AGM06, PU89b, TZ01], covers [ABCP96], dominating sets [DMP$^+$05], distance oracles [BK06, TZ05], emulators and distance preservers [BCE03, CE05], broadcasting [FPZW04], near-shortest path algorithms [Elk05, Elk07, EP04, EZ04]. Recent reviews of the literature on spanners can be found in [Pet07, Woo06]. Hence, understanding the properties of graph spanners and providing efficient algorithms for constructing them appear as a fundamental problem in distributed computing. However, sequential spanner construction algorithms, which aim at finding new stretch-size trade-offs, often use greedy techniques, whose adaptation to the distributed setting incurs an unavoidably high cost. This is mainly due to the symmetry breaking problems one has to deal with in a distributed environment. To overcome this difficulty, randomization is often used as an alternative tool, allowing one to obtain efficient distributed constructions of spanners whose quality is expected to be good. These randomized solutions have the disadvantage of being unacceptable for some applications that need strong guarantees on the properties of the constructed spanners.

In this paper, we describe a new *deterministic* distributed spanner construction algorithm, which is simultaneously optimal in size, stretch and construction time. Our algorithm is neither a derandomization of some known randomized one, nor an adaptation of some previous sequential techniques. Moreover, our solution is local in nature, since it does not require any global knowledge such as the size of the graph.

A particularly attractive feature of our algorithm is that, as shown later, it is *neighborhood optimal*, in the following sense. By the terminology of [KP98], given a problem P, let $\rho(P)$ be the radius (around each node) from which information must be fetched in order to solve P. An algorithm for P is *neighborhood optimal* if its time complexity is $O(\rho(P))$. For some problems, tight bounds on $\rho(P)$ are known and the challenge is to find neighborhood optimal solutions. For the graph spanner problem, neither nontrivial bounds of $\rho(P)$ nor optimal deterministic algorithms are known. In this paper, we both determine $\rho(P)$ precisely and provide a corresponding neighborhood optimal algorithm, thus making a step towards a better understanding of the locality of graph spanners.

1.2 Related Work

Given an n-node graph G and an integer parameter $k \geqslant 1$, it is now folklore that G admits a $(2k-1, 0)$-spanner with $O(n^{1+1/k})$ edges. The latter stretch-size trade-off is believed to be optimal according to the Erdös-Simonovits Conjecture proved for $k \in \{1, 2, 3, 5\}$ (see Section 3.2 for more details). The conjecture implies that (α, β)-spanners with $\alpha + \beta < 2k + 1$ require $\Omega(n^{1+1/k})$ edges in the worst-case. Recently, is has been proved in [Woo06] that $(1, \beta)$-spanners with $\beta < 2k$ requires indeed $\Omega(n^{1+1/k})$ edges in the worst-case, thus overcoming the dependence on the Erdös-Simonovits Conjecture.

There are mainly two classes of algorithms leading to the construction of high quality spanners. The first class is based on removing short cycles so that the graph is sufficiently sparse and yet has the required stretch. The second class is based on sparse partitions or

(d, c)-decompositions (cf. [Pel00]), which are efficient representations of the graph by weakly connected clusters (in terms of the number of inter-cluster edges or the cluster overlap) having small radius. Thus, using shortest path trees spanning the clusters, the nodes can be spanned using few edges while keeping the stretch low.

Most of the distributed algorithms providing high quality spanners are based on such sparse partitions or decompositions [ABCP96, ABCP98, Pel00]. At their best [ABCP98], these methods construct $(4k-3, 0)$-spanners with $O(n^{1+1/k})$ edges in $\Omega(n^{1/k+\epsilon})$ time where $\epsilon = \Omega(1/\sqrt{\log n}\,)$ is asymptotically better than $o(n)$. Straightforward (Las Vegas) randomized implementations lead to polylogarithmic expected time within the same stretch-size trade-off. Recently [DGP07], a new sequential algorithm for constructing size constrained spanners for bipartite graphs, together with a fast construction of $(O(\log n), O(\log n))$-decompositions using independent dominating sets, has enabled to break the sub-polynomial time barrier, providing $(4k-3, 0)$-spanners with $O(kn^{1+1/k})$ edges in $2^{O(k)} \log^{k-1} n$ time deterministically.

All of these (d, c)-decomposition based spanner algorithms rely more or less on the following idea: (1) find a subset of nodes X mutually at distance at least $2t + 1 \geqslant 3$ of each other; (2) in time t, build around each $x \in X$ a cluster consisting of its t-neighborhood; (3) in parallel, construct an efficient sparse spanner for each cluster; (4) span the inter-cluster edges. Each of these steps takes $O(t)$ distributed time except Step (1), which involves solving a symmetry breaking problem and thus incurs a high time cost in the deterministic setting. This difficulty can be solved by using a maximal independent set (MIS) in some t power of the graph G, or in time $O(\log n)$ if one is willing to increase t to $O(\log n)$ (provided that Step (4) is efficiently doable within $O(\log n)$ radii regions). Unfortunately, the distributed time complexity of MIS is a challenging question. Currently, the lower bound is $\Omega(\sqrt{\log n/\log\log n}\,)$ [KMW05]. Thus, any algorithm based on the previous technique will at least fail to break the sub-polylogarithmic barrier even when using randomization. However, other randomized algorithms using implicitly a kind of (d, c)-decomposition exist with faster running time but at the price of relaxing guarantees on the size of the spanners. A (Monte Carlo) algorithm that computes a $(2k-1, 0)$-spanner with expected size $O(kn^{1+1/k})$ in k time is described in [BS07]. The algorithm is based on sampling the nodes repeatidly with a given probability, constructing clusters around the sampled nodes, and then connecting the clusters. Let us remark that all the previous (deterministic or randomized) algorithms need to know the value of n or at least an upper bound on it.

Concerning the girth based algorithms, the only distributed result we know of was described in [DMP$^+$05], where it is shown how to construct a $(O(\log n), 0)$-spanner with $O(n)$ edges. Actually, we can generalize that algorithm for every parameter $k \geqslant 1$, obtaining spanners with $O(n^{1+1/k})$ edges in $n^{O(1/\sqrt{\log n})}$ time deterministically or $k \log^{O(1)} n$ using randomization. However the stretch is $(2k+1, 0)$ which is not optimal. The latter bound for the time complexity seems to be hard to improve since the use of a (d, c)-decomposition is a bottleneck to speeding up the construction.

Another kind of spanners are *additive* spanners, which are particularly interesting for approximating long distances. Only few constructions for pure additive spanners, i.e., for which the stretch is of the form $(1, \beta)$, are known, even in the sequential setting. In particular, sequential algorithms that construct a $(1, 2)$-spanner with $O(n^{3/2})$ edges and a $(1, 6)$-spanner with $O(n^{4/3})$ edges were given respectively in [EP04] and [BKMP05]. However, almost pure additive spanners have been described in some recent works. In [EP04, EZ04, Elk05], $(1+\epsilon, \beta)$-spanners with size $O(\beta n^{1+\delta})$ are constructed in $O(n^{\delta})$ time, where $\beta = \beta(\delta, \epsilon)$ is independent of n but grows super-polynomially in δ^{-1} and ϵ^{-1}. A sequential algorithm based on a randomized sampling technique was given in [TZ06], providing a spanner with $O(kn^{1+1/k})$ edges such that the distance d between any two nodes in the original graph is bounded by $d + o(d)$ in the spanner. Several variants have been developed recently in [Pet07]. Finally, specific distributed algorithms have been proposed in [DGP07] for $k = 2$. They provide spanners of $O(n^{3/2})$ edges with stretch $(1+\epsilon, 8\log n)$ and $(1+\epsilon, 4)$ in deterministic $O(\epsilon^{-1}\log n)$ time and $n^{O(1/\sqrt{\log n})} + O(\epsilon^{-1})$ time respectively. The $(1+\epsilon, 4)$-spanner algorithm is however shown to have a straightforward (Las Vegas) randomized implementation requiring $O(\epsilon^{-1} + \log n)$ time in expectation.

1.3 Our contributions

In Section 2, we present two deterministic distributed algorithms for constructing sparse low stretch spanners of every unweighted n-node graph.

- Given every $k \geqslant 1$, the first algorithm constructs in k rounds a $(2k-1, 0)$-spanner of $O(kn^{1+1/k})$ edges, thus attaining simultaneous optimality[1] in stretch, size and time. An important feature of our algorithm is that if n is not available to the nodes (which may often happen in concrete applications), then it executes in $3k-2$ rounds, and still returns a $(2k-1, 0)$-spanner with $O(kn^{1+1/k})$ edges.
- Based on this algorithm, we propose a second one that, for every $\epsilon > 0$, constructs a $(1+\epsilon, 2)$-spanner of $O(\epsilon^{-2}n^{3/2})$ edges in $O(\epsilon^{-1})$ rounds, without any prior knowledge on the graph.

As discussed earlier, best previous solutions achieving such near optimal stretch-size trade-off either make use of randomization and provide no performance guarantees [BS07], or require $\log^{\Omega(1)} n$ rounds [DGP07], and all require a priori knowledge of n.

In section 3, we present two lower bounds.

- Any (randomized) distributed algorithm requires k rounds in expectation to compute a $(2k-1, 0)$-spanner of $o(n^{1+1/(k-1)})$ edges for every $k \in \{2, 3, 5\}$. Assuming the Erdös-Simonovits Conjecture, the lower bound holds for every $k > 1$. According to our upper bound, k is optimal.
- It is also shown that for every $k > 1$, there is an $\epsilon > 0$ such that any (randomized) distributed algorithm that constructs a spanner with fewer than $n^{1+1/k+\epsilon}$ edges in at most n^{ϵ} expected rounds must stretch some distances by an additive factor of $n^{\Omega(\epsilon)}$. This result is in fact a corollary of a stronger technical result, saying that in t rounds distances in $tn^{\Theta(1/k^2)}$ must be stretched by a multiplicative factor of at least $1 + \Omega(k/t)$ if the spanner has fewer than $n^{1+1/k+\Theta(1/k^2)}$ edges.

We observe that both lower bounds hold even under the assumption that n is known to the nodes. For concreteness, consider the case $k = 2$. In this case, the first lower bound claims that two rounds are required to compute a $(3, 0)$-spanner with $o(n^2)$ edges. Our first algorithm returns a $(3, 0)$-spanner of $O(n^{3/2})$ edges in two rounds. Our second lower bound claims that,

[1] Optimality is subject to the Erdös-Simonovits Conjecture proved for $k \in \{1, 2, 3, 5\}$.

whereas $(1,2)$-spanners with $O(n^{3/2})$ edges exist for every graph, they cannot be computed distributively in less than $n^{1/4}$ time, or in other words, if running in, say, polylogarithmic time, any (randomized) distributed will stretch some distance by an additive factor of $n^{\Omega(1)}$. Actually, our lower bound shows that in t time the multiplicative stretch must be larger than $1+1/(t+1)$.

2 Deterministic Algorithms

2.1 The model

We consider unweighted n-node graphs. We assume the classical $\mathcal{LOCAL}$ model of computations (cf. [Pel00]), a.k.a. the free model [Lin92]. In this model, the nodes operate in synchronous discrete rounds. At each round, a node can send and/or receive messages of unbounded capacity to/from its neighbors and can perform any amount of local computations. Hence, each round costs one time unit. Also, nodes have unique identifiers that can be used for breaking symmetry. As long as we are concerned with running time and not with the cost of communication, synchronous and asynchronous message passing models are equivalent. Hereafter, unless stated explicitly, it is assumed that n is not known to nodes.

2.2 First algorithm

Let us consider a graph G. For every node u and integer r, $B_G(u,r)$ denotes the radius-r ball centered at u in G. When G is clear from the context we remove the subscript G from $B_G(u,r)$.

The algorithm, named $\textsc{Span}_k$ where $k \geqslant 1$ is an integral parameter, is described in Algorithm 1. It is performed on every node u in parallel. Roughly speaking, each node u maintains a region, i.e., a set $R(u)$ of nodes around it, which grows after each round. The current value of the next region for u is denoted by C, and the set of remaining uncovered neighbors is denoted by W. Initially, W contains all of u's neighbors. During the execution of the algorithm, some of these neighbors are gradually moved to the set L maintained by u. Actually, the set L contains the neighbors of u in the constructed spanner H. At the same time, some neighbors of u are discarded from W and will not be among u's neighbors in H. The strategy applied by the algorithm consists of covering those neighbors of u by the regions of the neighbors of u contained in the set L. At each round, L can grow by adding at most σ new nodes, where σ is a threshold value that cannot exceed $n^{1/k}$. If n is unknown, then u initializes σ to $|B(u,2k-1)|^{1/k}$ (computing this value at u requires $2k-1$ steps).

2.3 Correctness of Algorithm Span_k

Hereafter, unless stated explicitly, the stretch and size analysis of our spanners is assumed to be done independently in each connected components of the graph.

Let $L_i(u)$ (resp., $R_i(u)$) denote the set L (resp., $R(u)$) computed by u just after the ith round of algorithm $\textsc{Span}_k(u)$, and let $H_i(u)$ be the "star" subgraph of G composed of the edges connecting u to the nodes of $L_i(u)$. Let $H_i = \bigcup_{u \in V(G)} H_i(u)$.

The subgraph H_k is the spanner generated for G by algorithm $\textsc{Span}_k$. As we will see, H_k satisfies the desired properties. For that purpose we establish two key propositions.

$L := C := R(u) := \{u\}$, $W := B(u,1)$, $\sigma := |B(u,2k-1)|^{1/k}$
for $i := 1$ *to* k **do**
  Node u sends $R(u)$ to its neighbors, and receives $R(w)$ from all its neighbors w
  while $\exists w \in W$ *such that* $R(u) \cap R(w) \neq \varnothing$ **do**
    $W := W \setminus \{w\}$
  while $\exists w \in W$ *and* $|L| < i\sigma$ **do**
    $W := W \setminus \{v \in W \mid R(v) \cap R(w) \neq \varnothing\}$
    $L := L \cup \{w\}$
    $C := C \cup R(w)$
  $R(u) := C$

Algorithm 1: Algorithm $\text{SPAN}_k(u)$ for a node u. The set L computed by u is the set of the neighbors of u in the spanner.

Proposition 1 *At the end of round i, $R_i(u) \subseteq B_{H_i}(u,i)$.*

Proof. The proof is by induction. Let L_0, $R_0(u)$ and H_0 denote the initial values of L, $R(u)$ and H prior to round 1, i.e., let $H_0(u)$ be an empty graph on u and its neighbors and let $R_i(u) = \{u\}$, which is obviously included in $B_{H_0}(u,0) = \{u\}$. Consider the ith round. At the beginning of the round we have $R_{i-1}(u) \subseteq B_{H_{i-1}}(u,i-1)$. Also, $H_{i-1} \subseteq H_i$. Therefore, it suffices to show that every node which joins $R(u)$ during round i is also in $B_{H_i}(u,i)$. More precisely, for each $R_{i-1}(w)$ added to C, we need to show that the nodes of $R_{i-1}(w)$ join also $B_{H_i}(u,i)$. Observe that when the nodes of $R_{i-1}(w)$ join C, w is added to L. Thus, H_i contains the edge (u,w). It also includes H_{i-1}. As $R_{i-1}(w) \subseteq B_{H_{i-1}}(w,i-1)$ by the induction hypothesis, each node of $R_{i-1}(w)$ is reachable in H_i from u in at most i hops, so $R_{i-1}(w) \subseteq B_{H_i}(u,i)$. In summary, we get $R_i(u) \subseteq B_{H_i}(u,i)$ after round i. ∎

Corollary 1 *If $x \in R_i(u)$ at the end of round i then $d_{H_k}(u,x) \leqslant d_{H_i}(u,x) \leqslant i$.*

Proposition 2 *At the end of round i, one of the following two properties holds:*

(1) $|R_i(u)| \geqslant |B(u,2k-i)|^{i/k}$, or

(2) $R_i(v) \cap R_i(u) \neq \varnothing$ for all $v \in B(u,1)$.

Proof. We show the claim by induction on i. At the end of round 0 (i.e., just before round 1), $|R_0(u)| = 1$ for every u. Now consider round $i \geqslant 1$. By the induction hypothesis, at the beginning of round i, for each neighbor w of u in W, we have that either $|R_{i-1}(w)| \geqslant |B(w,2k-i+1)|^{(i-1)/k}$ or $R_{i-1}(v) \cap R_{i-1}(w) \neq \varnothing$ for all $v \in B(w,1)$. When $R_{i-1}(w) \cap R_{i-1}(u) \neq \varnothing$, the first while loop removes w from W. The nodes added to L in the second while loop must thus verify $|R_{i-1}(w)| \geqslant |B(w,2k-i+1)|^{(i-1)/k} \geqslant |B(u,2k-i)|^{(i-1)/k}$. If W is not emptied, $\sigma \geqslant |B(u,2k-i)|^{1/k}$ such nodes are added. Whenever such w is added, the second loop discards from W all nodes w' such that $R_{i-1}(w') \cap R_{i-1}(w) \neq \varnothing$, and moreover, these sets do not intersect $R_{i-1}(u)$. Hence the size of C increases by at least $|B(u,2k-i)|^{i/k}$. If the loops stop because $W = \varnothing$, then we have $R_i(v) \cap R_i(u) \neq \varnothing$ for all

$v \in B(u, 1)$. ∎

Notice that if property (2) of the proposition holds after some round i, then the set W is empty at that stage, and the set $R(u)$ will not change any more, so property (2) will continue to hold throughout the remainder of the execution.

We also remark that Proposition 2 and Proposition 1 imply that after the kth round, $R(v) \cap R(u) \neq \varnothing$ for all adjacent nodes u, v.

We are now ready to prove the following.

Theorem 1 *Algorithm* $\textsc{Span}_k$ *is a deterministic distributed algorithm that for every* n*-node graph* G *computes a* $(2k-1, 0)$*-spanner of size at most* $kn^{1+1/k}$ *of* G *in time* k*. Moreover, if* n *is unknown, then the algorithm requires time* $3k-2$*.*

Proof. Let $H = H_k$ denote the subgraph obtained at the end of the k rounds. The number of edges incident to u that are selected to H after the k rounds of algorithm $\textsc{Span}_k(u)$ is $|L_k(u)| \leqslant k\sigma$, which is at most $kn^{1/k}$. Hence overall, H has at most $kn^{1+1/k}$ edges.

Let us now analyze the stretch of H. Consider two neighbors u and v in G. By the above two propositions, we have $R(u) \subseteq B_H(u, k)$ and $R(v) \subseteq B_H(v, k)$ and there exists a node $x \in R(u) \cap R(v)$. We thus get $d_H(u, v) \leqslant 2k$. To prove the slightly better bound of $2k-1$, we look more carefully at the round i in which u has removed v from its set W of uncovered neighbors. There are two cases to consider. In the first case, v has been removed according to the first while loop. In that case, we have $d_{H_{i-1}}(u, v) \leqslant 2(i-1) \leqslant 2(k-1)$ by Proposition 1. In the second case, v has been removed according to the second while loop. Let w be the node added to L_i in that step. This node satisfies $R(v) \cap R(w) \neq \varnothing$. Let x be any node in $R(w) \cap R(v)$ (possibly $x = v = w$ or $x = v$). Relying on the fact that $d_H(u, w) = 1$, and that $d_{H_{i-1}}(w, x) \leqslant i-1$ and $d_{H_{i-1}}(v, x) \leqslant i-1$ by Cor. 1, we obtain that $d_H(u, v) \leqslant 1 + 2(k-1) = 2k-1$. This completes the stretch analysis.

Let us now analyze the time complexity. Clearly, once the initializations are done, $\textsc{Span}_k$ performs in at most k rounds. Let us first observe that the initialization of the sets W can be done on the fly after the first communication step. We observe also that Proposition 2 holds for every value of $\sigma \in [|B(u, 2k-1)|^{1/k}, n^{1/k}]$. So, if the value of n is known to the node u, then σ can be set to $n^{1/k}$ without communication. Otherwise, σ can be set to the value $|B(u, 2k-1)|^{1/k}$, which can be computed locally at u in $2k-2$ extra rounds (one round can be saved by combining the ball size computation with the first round). ∎

2.4 Improving the stretch

In this section we show how to extend algorithm $\textsc{Span}_2$ to obtain almost pure additive stretch spanners. The extended algorithm, denoted by $\textsc{Span}_{2,t}$, where $t \geqslant 2$ is an integer, is described in a high level way in Algorithm 2. For clarity we have omitted some communication details. For $\epsilon = \Theta(1/t)$, we prove the following theorem.

Theorem 2 *For every* $\epsilon \in (0, 2]$*, there exists a deterministic distributed algorithm that for every* n*-node graph (where* n *is unknown to the nodes) computes a* $(1+\epsilon, 2)$*-spanner with* $O(\epsilon^{-2}n^{3/2})$ *in* $O(\epsilon^{-1})$ *time.*

$R(u) := \{u\}$, $r := \lfloor t/2 \rfloor$, $\sigma := |B(u, 2r+1)|^{1/2}$
Node u sends $R(u)$ to its neighbors, and receives $R(w)$ from all its neighbors w
Node u chooses $\min\{\deg(u), \lceil\sigma\rceil - 1\}$ neighbors and add them to set $R(u)$
$H_u := (R(u), \bigcup_{w \in R(u)\setminus\{u\}} \{u, w\})$
if $R(u) = B(u, 1)$ **then** $F_u :=$ true **else** $F_u :=$ false
Node u sends $(R(u), F_u)$ to all nodes in $B(u, r)$ and receives $(R(w), F_w)$ from every $w \in B(u, r)$
for $\ell := 1$ *to* r **do**
    $W_\ell := \{w \in B(u, \ell) \mid F_w \text{ is false}\}$
    while $\exists w \in W_\ell$ **do**
        $W_\ell := W_\ell \setminus \{v \mid R(v) \cap R(w) \neq \varnothing\}$
        add to H_u a shortest path in G from u to w

Algorithm 2: Algorithm $\textsc{Span}_{2,t}(u)$ for a node u with parameter t. The spanner is the union of all subgraphs H_u computed by all nodes u.

Proof. First we analyze the time complexity of algorithm $\textsc{Span}_{2,t}$. The initialization steps required before the for loop can be done on the fly in $2r + 1 + r$ rounds. The for loop can be implemented in r rounds. In fact, observe that if node u knows the topology of its r-ball $B(u, r)$, then it can compute locally all the shortest paths that must be added to the spanner, then broadcast the result up to distance r at the end of the for loop. Overall, algorithm $\textsc{Span}_{2,t}$ requires at most $2t + 1$ rounds. We remark that if n is known then we can save t rounds thus obtaining $t + 1$ time complexity.

Let us now analyze the properties of the spanner $H = \cup_{u \in V(G)} H_u$. Consider a node u in $V(G)$. It is clear that before the for loop, u adds at most $\sigma \leqslant n^{1/k}$ edges to the spanner. Moreover, either F_u is true and H_u contains all edges linking to u or $|R(u)| \geqslant \sigma$. Now, consider an iteration ℓ of the for loop. Each $w \in W_\ell$ has F_w set to false and satisfies $|R(w)| \geqslant |B(w, 2r+1)|^{1/2}$. As $d_G(u, w) = \ell \leqslant r$, $|R(w)| \geqslant |B(u, r+1)|^{1/2}$. We remark that if a node $v \in R(w)$ then $R(v) \cap R(w) \neq \varnothing$. Thus, the entire set $R(w)$ is removed from W_ℓ. The sets $R(w)$ removed from W_ℓ are disjoint and their union lies in $B(u, \ell + 1)$ as $R(w)$ contains neighbors of w. We can thus iterate at most $|B(u, r+1)|^{1/2}$ times before W_ℓ is empty. Therefore, node u adds at most $n^{1/2}$ paths of ℓ edges to H_u at iteration ℓ. Overall, the computed spanner has at most $O(r^2 \cdot n^{1/2} \cdot n) = O(t^2 \cdot n^{3/2})$ edges.

Consider the stretch obtained for some v with $d_G(u, v) = \ell \leqslant r$. We show by induction on ℓ that $d_H(u, v) \leqslant \ell + 2$. This is clearly true for $\ell = 0$. Consider a node v removed from W_ℓ during iteration $\ell \geqslant 1$ of the for loop. If a path from u to v was added to H_u then $d_H(u, v) = \ell$. If $R(v) \cap R(w) \neq \varnothing$ for some w selected in W_ℓ during the while loop, then $d_H(u, v) \leqslant \ell + 2$. Finally, if v was removed because F_v was set to true, all neighbors of v are in H. Let v' be a neighbor such that $d_G(u, v') = \ell - 1$. We deduce by induction that $d_H(u, v') \leqslant \ell + 1$ and thus $d_H(u, v) \leqslant \ell + 2$. For a node v at any distance $d = d_G(u, v)$, we can then prove by induction that $d_H(u, v) \leqslant \left(1 + \frac{2}{r}\right) d + 2$. We already know it is true for $d \leqslant r$. For $d > r$, consider the node v' at distance r from u in a shortest path from u to v in G. We get

$$\begin{aligned} d_H(u,v) &\leqslant d_H(u,v') + d_H(v',v) \leqslant r + 2 + \left(1 + \frac{2}{r}\right) d_G(v',v) + 2 \\ &\leqslant \frac{r+2}{r} r + \left(1 + \frac{2}{r}\right)(d-r) + 2 \\ &\leqslant \left(1 + \frac{2}{r}\right) d + 2 \ . \end{aligned}$$

As $r = \lfloor t/2 \rfloor > \frac{1}{2}(t-1)$ we obtain stretch $\left(1 + \frac{4}{t-1}, 2\right)$. Because r must be $\geqslant 1$, we have $t \geqslant 2$. Theorem 2 follows by letting $\epsilon = 4/(t-1) \in (0,4]$ for every $t \geqslant 2$. ∎

3 Lower Bounds

For convenience, let us say that a distributed algorithm A computes a spanner H for G if after running A, for every edge (u,v) of H, either u or v is aware of this edge being in H. Clearly, at the cost of one additional communication round, one can guarantee that both endpoints are aware of the edge.

For proving our results we make use of (a variant of) a classical lower bounding technique, used for instance in [KMW05]. A brief description of this technique is presented in the next section.

3.1 Equivalent views

Let $G(V,E)$ be an n-node graph with distinct integer node identities $\mathrm{ID} : V \mapsto \{1,\ldots,n\}$, and let $t \in \mathbb{N}$. A *decoration* is a node labeling ξ that associates with each node u of G an integer sequence $\xi(u) = (r_1, r_2, \ldots)$ where $r_1 = \mathrm{ID}(u)$. Let us denote by $\xi(u,t) = (r_1,\ldots,r_t)$ the length-t prefix of $\xi(u)$. The *view* of u, denoted by $\textsc{View}(u,t,G,\xi)$, is the subgraph of G induced by $B_G(u,t)$ in which edges between nodes at distance t from u are removed, and in which each node v is labeled with the sequence $\xi(v, t+1-d_G(u,v))$. Informally, $\textsc{View}(u,t,G,\xi)$ is the total information available at u after t rounds, ID's and random choices are captured by the decoration ξ of the graph.

A round consists of two steps: a communication step between neighbors (send/receive statements) followed by a local computation step, including a random choice considered as an integer. Consider a run of a randomized algorithm A on G. The state of node u after t rounds of A can be expressed by a function Φ_A depending on (1) the partial topology of G received by u after t communication steps, and (2) all ID's and random choices made by the nodes of $B_G(u,t)$. More precisely, after t rounds u is aware of at most $t - d_G(u,v)$ first[2] random choices of v.

In the following we identify a *run* of A by a decoration of the graph, representing the ID's and the list of random choices made by all nodes running A. As a result, the state of u after t

[2]In our model, after the first round u is aware of its own random choice only, and of its neighbors ID's. The result of the first local computation of any u neighbor becomes available to u only at the beginning of the second round.

rounds of a run ξ of A on G can be defined as a function, denoted by Φ_A, of $\textsc{View}(u,t,G,\xi)$, which is nothing else than a subgraph of G whose nodes are labeled by some prefixes of ξ. Note that the mapping Φ_A itself may depends on a priori knowledge, like the number of nodes of G, but is independent of ID's and random choices which are contained in the views.

Now, assume there are two graphs G_1, G_2 with two nodes $u_1 \in V(G_1)$, $u_2 \in V(G_2)$ such that, in order to solve a problem P on the family $\{G_1, G_2\}$, algorithm A has to output a different state for u_1 and for u_2. For instance, assume P is the problem of constructing a spanning tree on a graph family including n-node paths and n-node cycles. Let us fix t, and assume that for every run ξ of A, $\textsc{View}(u_1,t,G_1,\xi) = \textsc{View}(u_2,t,G_2,\xi)$. Note that in particular, $\mathrm{ID}(u_1) = \mathrm{ID}(u_2)$. Then, applying Φ_A on both sides, we conclude that A does not solve P in time t, because the state will be the same in u_1 and in u_2 whereas different outputs are required. If A solves P for all instances, then its time complexity is thus strictly greater than t. Actually, even the *expected* time of A is strictly greater than t, since the property holds for every run of A, hence it holds for the average as well. Coming back to our concrete example of spanning tree problem P, we deduce[3] that any randomized distributed algorithm solving P has expected time at least $n/2$.

3.2 Dense large girth graphs

Both our two lower bounds rely on the existence of dense graphs with large *girth*, that is, with large minimum cycle length. It is known that the complete bipartite graph $K_{n/2,n/2}$ is a girth 4 graph with maximum edge density: it has $n^2/4$ edges. Actually, there are constructions of n-node graphs with girth at least $2k+2$ and $\Omega(n^{1+1/k})$ edges, for $k \in \{1,2,3,5\}$ (see [Ben66, Wen91]). According to the Erdös-Simonovits [ES82] Conjecture (see also [Erd64, BS74, KKS98]), the result must hold for every k. As constructed in [LUW95], for every $k \geqslant 1$, there are graphs of girth $2k+2$ with $\Omega(n^{1+\frac{2}{3k}})$ edges.

Observe that if G_g is a girth g graph with maximum number of edges, then it is connected[4].

We are interested particularly in *bipartite* dense graphs of high girth. For every n and $1 \leqslant k \leqslant \log n$, consider the class of connected bipartite n-node graphs with girth at least $2k+2$, and pick $\mathcal{B}(n,k)$ to be (some) graph of maximum number of edges in this class.

We rely on the following well-known simple observation.

Observation 1 *Every m-edge graph contains a bipartite subgraph with at least $m/2$ edges.*

The following proposition is derived from the above discussion and observation. Let

$$\Psi(k) = \begin{cases} k, & k \in \{1,2,3,5\}, \\ 3k/2, & \text{otherwise.} \end{cases}$$

Lemma 1 *For all k,n such $1 \leqslant k \leqslant \log n$, the graph $\mathcal{B}(n,k)$ has at least $c_0 n^{1+1/\Psi(k)}$ edges for constant $c_0 > 0$. Moreover, assuming the Erdös-Simonovits Conjecture, $\Psi(k) = k$ for every k.*

[3]First argue that in the cycle, there must exists a node u_1 whose state corresponds to the removal of an edge. Then construct a decoration ξ on the path (including an ID assignment) such that, for u_1 and for the node u_2 of maximal eccentricity in the path, both views are the same.

[4]Otherwise, if G_g is composed of $c \geqslant 2$ connected components, one can add $c-1$ edges between them to yield a denser (connected) girth g graph.

3.3 A time lower bound

We prove that k rounds steps are necessary in expectation for distributively constructing sparse $(2k-1,0)$-spanners. Algorithm $\textsc{Span}_k$ shows that k rounds suffice.

A similar lower bound claimed in [Elk07] appears in [Elk06][5]. For completeness, and in order to illustrate the lower bound technique, we present below a variant of the proof presented in [Elk06].

Note that in the following statement, the algorithm may be dependent on n and k, hence the lower bound holds even if nodes are aware of these values.

Proposition 3 *Let n,k be integers such that $1 < k \leqslant \log n$. Consider a randomized distributed algorithm A that for every connected n-node graph computes a $(2k-1,0)$-spanner with fewer than $c_0 n^{1+1/\Psi(k-1)}$ edges. Then A has expected time at least k.*

Proof. Let t_G be the minimum execution time of A on the graph G, taken over all random choices made by A. Let t be $\max\{t_G\}$ over all connected n-node graphs G. Note that t lower bounds the expected time of A.

Consider $\mathcal{B}(n,k-1)$, $k>1$. Since its girth is $2(k-1)+2=2k$, in every node u, the ball $B_{\mathcal{B}(n,k-1)}(u,k-1)$ is isomorphic to a tree of depth $k-1$. Let $\mathcal{T}$ be the (infinite) set of all decorated n-node trees, i.e., trees whose nodes u are labeled by integer sequences $\xi(u)$.

Assume that $t \leqslant k-1$. For every node u and run ξ of A, there exists a tree $T \in \mathcal{T}$ such that $\textsc{View}(u,t,\mathcal{B}(n,k-1),\xi) = \textsc{View}(u,t,T,\xi)$, the number of nodes of both $\mathcal{B}(n,k-1)$ and T is n and it is known to u. Indeed, T can be chosen as a shortest path tree rooted at u in $\mathcal{B}(n,k-1)$, and labeled by ξ. Thus, after t rounds, the state in u computed by Φ_A is the same in $\mathcal{B}(n,k-1)$ and in T. If none of the nodes decides to remove one of its incident edges, then the number of edges of the spanner for $\mathcal{B}(n,k-1)$ is $c_0 n^{1+1/\Psi(k-1)}$. This contradicts the sparsity guarantee of A. Thus, there exists a run ξ_0 and a node u_0 in $\mathcal{B}(n,k-1)$ that decides by Φ_A to remove an edge. Selecting the tree T_0 such that $\textsc{View}(u_0,t,\mathcal{B}(n,k-1),\xi_0) = \textsc{View}(u_0,t,T_0,\xi_0)$ shows that A does not provide a spanner for T_0, since the spanner must be connected: a contradiction. Therefore $t > k-1$. ∎

3.4 A stretch lower bound

Theorem 3 *Let k,ϵ be such that $1<k\leqslant \log n$, and $1/\epsilon > 3k^2-k$. Consider any randomized distributed algorithm A that for every connected n-node graph computes a spanner with fewer than $\lambda \cdot n^{1+1/\Psi(k)}$ edges in t expected time, for $k-3 \leqslant t \leqslant n^\epsilon$ and $\lambda \leqslant n^\epsilon$. Then there exist a graph G and two nodes u,v at distance $d_G(u,v) = \Omega(tn^\epsilon)$ such that in the spanner H computed by A,*

$$d_H(u,v) \;>\; \left(1+\frac{k-1}{t+1}\right)\cdot d_G(u,v) + 2k-3\ .$$

Before presenting the proof, we note that Theorem 3 yields an interesting corollary, which is independent of the Erdös-Simonovits Conjecture.

[5]The bound proved in [Elk06][Theorem 10.2, page 70] is slightly different, despite proof similarities. E.g., for $k=2$, it gives at least $k-1=1$ round for computing a $(3,0)$-spanner of at most $n^{3/2}\log^{O(1)} n$ edges, whereas our statement claims a matching lower bound of $k=2$ rounds for a $(3,0)$-spanner of $o(n^2)$ edges.

Corollary 2 *Every randomized distributed algorithm that for every n-node graph computes an (α, β)-spanner with fewer than $n^{1+1/k+1/(3k^2)}$ edges in $t \leqslant n^{1/(3k^2)}$ expected time, for every $k > 1$, must verify $\alpha = 1 + \Omega(k/t)$, or $\beta = \Omega(kn^{1/(3k^2)})$.*

Proof. Theorem 3 with $\epsilon = 1/(3k^2)$ and $\lambda = n^\epsilon$ implies that there are distance $d_G(u,v) = \Theta(tn^\epsilon)$ such that:

$$\begin{aligned} d_H(u,v) &> \left(1 + \frac{k-1}{t+1}\right) \cdot d_G(u,v) + \Omega(k) \\ &= \left(1 + \frac{k-1}{2(t+1)}\right) \cdot d_G(u,v) + \frac{k-1}{2(t+1)} \cdot d_G(u,v) + \Omega(k) \\ &= \left(1 + \frac{k-1}{2(t+1)}\right) \cdot d_G(u,v) + \Omega(kn^\epsilon) \end{aligned}$$

that implies $\alpha = 1 + \Omega(k/t)$, or $\beta = \Omega(kn^\epsilon)$. ∎

Proof of Theorem 3 Let $B(X,Y,E)$ be a p-node bipartite graph, hereafter called the *base-graph*, whose parts are $X = \{x_1, x_2, \ldots\}$ and $Y = \{y_1, y_2, \ldots\}$. For every sequence S of b edges of B (possibly with repetitions), $S = \langle (x_{j_1}, y_{j_1}), \ldots, (x_{j_b}, y_{j_b}) \rangle$, we construct a graph $P(S)$, called a *block-path*, as follows (see Fig. 1 for an example):

1. $P(S)$ is composed of b disjoint copies of B denoted by $B^{(i)}(X^{(i)}, Y^{(i)}, E^{(i)})$ for $i = 1, \ldots, b$, each referred to as a *block* of $P(S)$.
2. In each block $B^{(i)}$, we attach to each node $x_\ell^{(i)} \in X^{(i)}$ (resp., $y_\ell^{(i)} \in Y^{(i)}$) a (distinct) new length-t *x-leg* (resp., *y-leg*) path whose other endpoint is named $\tilde{x}_\ell^{(i)}$ (resp., $\tilde{y}_\ell^{(i)}$.
3. Every two consecutive blocks $B^{(i)}$ and $B^{(i+1)}$, for $i \in \{1, \ldots, b-1\}$, are connected by a single *inter-block* edge, between $\tilde{y}_{j_i}^{(i)}$ and $\tilde{x}_{j_{i+1}}^{(i+1)}$.

Note that any two consecutive blocks $B^{(i)}, B^{(i+1)}$ in the block-path are connected by a unique path of length $2t+1$ via the single inter-block edge which connects the endpoints of the ith and $(i+1)$st edges of S. Note also that if B is connected, then so is the block-path.

For every block-path $P(S)$, and every $\beta \leqslant b$, there is a (shortest) path from $x_{j_1}^{(1)}$ to $y_{j_\beta}^{(\beta)}$ of the form

$$(\ x_{j_1}^{(1)}, y_{j_1}^{(1)}, \tilde{y}_{j_1}^{(1)}, \tilde{x}_{j_2}^{(2)}, x_{j_2}^{(2)}, y_{j_2}^{(2)}, \tilde{y}_{j_2}^{(2)}, \tilde{x}_{j_3}^{(3)}, \ldots, x_{j_\beta}^{(\beta)}, y_{j_\beta}^{(\beta)}\ )$$

that alternates successively between an edge of $B^{(i)}$, a length-t y-leg path, an inter-block edge and a length-t x-leg path. The total length of this path is

$$d_{P(S)}(x_{j_1}^{(1)}, y_{j_\beta}^{(\beta)}) \ = \ (2t+2)(\beta - 1) + 1 \ . \tag{1}$$

We now fix the base-graph $B = \mathcal{B}(p, k-1)$, where $k > 1$. For convenience, let $h(k) = 1 + 1/\Psi(k)$. The number of nodes of $P(S)$ is $bpt = n$. Let $b = n^\epsilon$.

The edge set of $P(S)$ is composed of $c_0 p^{h(k-1)}$ block edges and pt path-edges for each of the b blocks, and $b-1$ inter-block edges. The total number of edges of $P(S)$ is thus

$$m \ = \ b\left(c_0 p^{h(k-1)} + pt\right) + b - 1 \ > \ c_0 b p^{h(k-1)} \ . \tag{2}$$

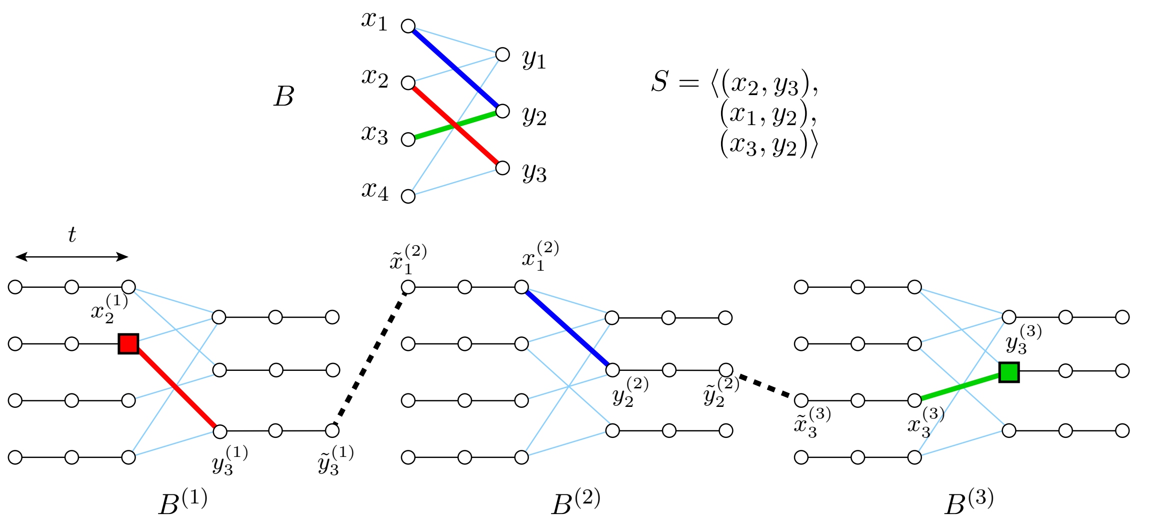

Figure 1: A block-path $P(S)$ with base-graph B for $b = 3$, $p = 7$ and $t = 2$. Inter-block edges are dashed.

Assume that A is a randomized distributed algorithm computing, for every connected n-node graph, a spanner with fewer than $\lambda \cdot n^{h(k)}$ edges in t rounds in expectation. Consider any run ξ of A on some block-path $P(S)$. Observe that in constructing the spanner, A can only remove edges inside blocks, as all x/y-leg path edges and inter-block edges are bridges. Let r_i be the number of edges removed by A in block $B^{(i)}$. In order to get the right sparsity, we must have $\sum_{i=1}^{b} r_i \geqslant m - \lambda n^{h(k)}$. Since each r_i is upper bounded by the number of edges of a block, i.e., $c_0 p^{h(k-1)}$, it follows that the number of blocks $B^{(i)}$ for which $r_i \geqslant 1$ is at least

$$\beta \;=\; \frac{m - \lambda n^{h(k)}}{c_0 p^{h(k-1)}} \,.$$

One can verify that, for $\Psi(k) = ck$ and sufficiently large n,

$$\begin{aligned} 1/\epsilon > 2ck^2 - 2k(c-1) \;\;\Leftrightarrow\;\; h(k) \;&<\; (1-2\epsilon)h(k-1) \\ \Rightarrow\;\; n^{h(k)} \;&<\; \frac{1}{2} c_0 n^{(1-2\epsilon)h(k-1)} \,. \end{aligned} \tag{3}$$

In particular, for $c = 3/2$, the condition $1/\epsilon > 3k^2 - k$ implies Eq. (3). Since $\lambda \leqslant n^{\epsilon} = b$, and since $p \geqslant n^{1-2\epsilon}$ (as $bt \leqslant n^{2\epsilon}$), Eq. (3) yields

$$\lambda \cdot n^{h(k)} \;\leqslant\; \frac{1}{2} c_0 b n^{(1-2\epsilon)h(k-1)} \;\leqslant\; \frac{1}{2} c_0 b p^{h(k-1)} \,.$$

By Eq. (2),

$$m - \lambda n^{h(k)} \;\geqslant\; \frac{1}{2} c_0 b p^{h(k-1)} \,.$$

If follows that

$$\beta \;=\; \frac{\frac{1}{2} c_0 b p^{h(k-1)}}{c_0 p^{h(k-1)}} \;=\; \frac{b}{2} \,.$$

Thus for every ξ there exist β blocks $B^{(b_1)}, \ldots, B^{(b_\beta)}$, each with a node $z_{j_i}^{(i)} \in X^{(i)} \cup Y^{(i)}$, such that $\Phi_A(\textsc{View}(z_{j_i}^{(i)}, t, P(S), \xi))$ is a state of $z_{j_i}^{(i)}$ corresponding to the removal of at least one edge of $B^{(b_i)}$ incident to $z_{j_i}^{(i)}$. Let $\mathcal{V}_{S,\xi}^{(i)} = \textsc{View}(z_{j_i}^{(i)}, t, P(S), \xi)$.

Unfortunately, the views $\mathcal{V}_{S,\xi}^{(i)}$ (for different i and fixed S, ξ) are pairwise different for at least two reasons: (1) the base-graph B may be not vertex-transitive; and (2) nodes $z_{j_i}^{(i)}$ are taken from the same graph. Thus their ID's differ, and thus their views do.

Consider the set $\mathcal{P}$ of all decorated block-paths $P(S)$, for all the sequences S composed of b edges of the base-graph B. Consider $\mathcal{V}_{S,\xi} = \bigcup_{i=1}^{\beta} \mathcal{V}_{S,\xi}^{(i)}$, composed of the β views defined as above. Because no view of any $z_{j_i}^{(i)}$ contains an inter-block edge, $\mathcal{V}_{S,\xi}$ is independent of S. In other words, for every ξ, $\mathcal{V}_{S,\xi} = \mathcal{V}_{S',\xi} = \mathcal{V}_\xi$ for all S, S'.

From the above discussion, it follows that for all pairs (S, ξ), there is a decorated graph $P_\xi \in \mathcal{P}$ isomorphic to $P(S)$ such that $b_i = i$ for every $i \in \{1, \ldots, \beta\}$, i.e., that such that the blocks with at least one edge removed appear consecutively. For $i \in \{1, \ldots, \beta\}$, pick $(x_{j_i}^{(i)}, y_{j_i}^{(i)})$ to be one of the edges removed in block $B^{(i)}$. We simply set S_ξ as the following edge sequence of B: $S_\xi = \big\langle (x_{j_1}, y_{j_1}), \ldots, (x_{j_\beta}, y_{j_\beta}) \big\rangle$.

To summarize, for each run ξ of A, we have constructed a block-path $P(S_\xi)$ where all the edges corresponding to S_ξ are removed by A (because the view $\mathcal{V}_\xi$ is independent on S_ξ). As the girth of $B^{(i)}$ is at least $2k$, the distance in the spanner H_ξ returned by A on run ξ between $x_{j_i}^{(i)}$ and $y_{j_i}^{(i)}$ is at least $2k-1$. Because all the edges of $P(S_\xi)$, except the edges of the blocks, are bridges, it follows that

$$\begin{aligned} d_H = d_{H_\xi}(x_{j_1}^{(1)}, y_{j_\beta}^{(\beta)}) &\geqslant (2t+1+2k-1)(\beta-1) + 2k - 1 \\ &= (2t+2)(\beta-1) + 1 + (2k-2)\beta \ . \end{aligned}$$

By Eq. (1), we have that

$$d_G \ = \ d_{P(S_\xi)}(x_{j_1}^{(1)}, y_{j_\beta}^{(\beta)}) \ = \ (2t+2)(\beta-1)+1 \ .$$

Hence $\beta = (d_G - 1)/(2t+2) + 1$. Therefore,

$$\begin{aligned} d_H &\geqslant d_G + (2k-2)\beta \\ &= \left(1 + \frac{k-1}{t+1}\right) d_G + 2k - 2 - \frac{k-1}{t+1} \\ &> \left(1 + \frac{k-1}{t+1}\right) d_G + 2k - 3 \end{aligned}$$

because for $t > k-2$, $(k-1)/(t+1) < 1$. We have $d_G = \Theta(tb) = \Omega(tn^\epsilon)$. The above lower bound on the distance in the spanner output by A after t rounds is for every ξ, thus it holds for the expectation. This complete the proof. ∎

References

[ABCP96] Baruch Awerbuch, Bonnie Berger, Lenore Jennifer Cowen, and David Peleg. Fast distributed network decompositions and covers. *Journal of Parallel and Distributed Computing*, 39(2):105–114, 1996.

[ABCP98] Baruch Awerbuch, Bonnie Berger, Lenore Jennifer Cowen, and David Peleg. Near-linear time construction of sparse neighborhood covers. *SIAM Journal on Computing*, 28(1):263–277, February 1998.

[AGM06] Ittai Abraham, Cyril Gavoille, and Dahlia Malkhi. On space-stretch trade-offs: Upper bounds. In 18^{th} *Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA)*, pages 207–216. ACM Press, July 2006.

[Awe85] Baruch Awerbuch. Complexity of network synchronization. *Journal of the ACM*, 32(4):804–823, 1985.

[BCE03] Béla Bollobás, Don Coppersmith, and Michael Elkin. Sparse distance preservers and additive spanners. In 14^{th} *Symposium on Discrete Algorithms (SODA)*, pages 414–423. ACM-SIAM, January 2003.

[Ben66] C. T. Benson. Minimal regular graphs of girth eight and twelve. *Canadian Journal of Mathematics*, 18:1091–1094, 1966.

[BK06] Surender Baswana and Telikepalli Kavitha. Faster algorithms for approximate distance oracles and all-pairs small stretch paths. In 47^{th} *Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 591–602. IEEE Computer Society Press, October 2006.

[BKMP05] Surender Baswana, Telikepalli Kavitha, Kurt Mehlhorn, and Seth Pettie. New constructions of (α, β)-spanners and purely additive spanners. In 16^{th} *Symposium on Discrete Algorithms (SODA)*, pages 672–681. ACM-SIAM, January 2005.

[BS74] J. Adrian Bondy and M. Simonovits. Cycle of even length in graphs. *Journal of Combinatorial Theory, Series B*, 16(2):97–105, 1974.

[BS07] Surender Baswana and Sandeep Sen. A simple and linear time randomized algorithm for computing sparse spanners in weighted graphs. *Random Structures and Algorithms*, 30(4):532–563, July 2007.

[CE05] Don Coppersmith and Michael Elkin. Sparse source-wise and pair-wise distance preservers. In 16^{th} *Symposium on Discrete Algorithms (SODA)*, pages 660–669. ACM-SIAM, January 2005.

[DGP07] Bilel Derbel, Cyril Gavoille, and David Peleg. Deterministic distributed construction of linear stretch spanners in polylogarithmic time. In 21^{st} *International Symposium on Distributed Computing (DISC)*, volume 4731 of Lecture Notes in Computer Science, pages 179–192. Springer, September 2007.

[DMP$^+$05] Devdatt Dubhashi, Alessandro Mai, Alessandro Panconesi, Jaikumar Radhakrishnan, and Aravind Srinivasan. Fast distributed algorithms for (weakly) connected dominating sets and linear-size skeletons. *Journal of Computer and System Sciences*, 71:467–479, 2005.

[Elk05] Michael Elkin. Computing almost shortest paths. *ACM Transactions on Algorithms*, 1(2):283–323, October 2005.

[Elk06] Michael Elkin. A near-optimal fully dynamic distributed algorithm for maintaining sparse spanners. Technical report, arXiv:cs.DS/0611001v1, November 2006.

[Elk07] Michael Elkin. A near-optimal fully dynamic distributed algorithm for maintaining sparse spanners. In 26^{th} *Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pages 195–204. ACM Press, August 2007.

[EP04] Michael Elkin and David Peleg. $(1+\epsilon, \beta)$-spanner constructions for general graphs. *SIAM Journal on Computing*, 33(3):608–631, 2004.

[Erd64] Paul Erdös. Extremal problems in graph theory. In *Publ. House Cszechoslovak Acad. Sci., Prague*, pages 29–36, 1964.

[ES82] Paul Erdös and M Simonovits. Compactness results in extremal graph theory. *Combinatorica*, 2(3):275–288, 1982.

[EZ04] Michael Elkin and Jian Zhang. Efficient algorithms for constructing $(1+\epsilon, \beta)$-spanners in the distributed and streaming models. In 23^{rd} *Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pages 160–168. ACM Press, July 2004.

[FPZW04] Arthur M. Farley, Andrzej Proskurowski, Daniel Zappala, and Kurt Windisch. Spanners and message distribution in networks. *Discrete Applied Mathematics*, 137(2):159–171, 2004.

[KKS98] Y. Kohayakawa, B. Kreuter, and A. Steger. An extremal problem for random graphs and the number of graphs with large even-girth. *Combinatorica*, 18(1):101–120, 1998.

[KMW05] Fabian Kuhn, Thomas Moscibroda, and Roger Wattenhofer. On the locality of bounded growth. In 24^{th} *Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pages 60–68. ACM Press, July 2005.

[KP98] Shay Kutten and David Peleg. Fast distributed construction of small k-dominating sets and applications. *Journal of Algorithms*, 28(1):40–66, 1998.

[Lin92] Nathan Linial. Locality in distributed graphs algorithms. *SIAM Journal on Computing*, 21(1):193–201, 1992.

[LUW95] Felix Lazebnik, Vasiliy A. Ustimenko, and Andrew J. Woldar. A new series of dense graphs of high girth. *Bulletin of the American Mathematical Society (New Series)*, 32(1):73–79, January 1995.

[Pel00] David Peleg. *Distributed Computing: A Locality-Sensitive Approach.* SIAM Monographs on Discrete Mathematics and Applications, 2000.

[Pet07] Seth Pettie. Low distortion spanners. In 34^{th} *International Colloquium on Automata, Languages and Programming (ICALP)*, volume 4596 of Lecture Notes in Computer Science, pages 78–89. Springer, July 2007.

[PU89a] David Peleg and Jeffrey D. Ullman. An optimal synchornizer for the hypercube. *SIAM Journal on Computing*, 18(4):740–747, 1989.

[PU89b] David Peleg and Eli Upfal. A trade-off between space and efficiency for routing tables. *Journal of the ACM*, 36(3):510–530, July 1989.

[TZ01] Mikkel Thorup and Uri Zwick. Compact routing schemes. In 13^{th} *Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA)*, pages 1–10. ACM Press, July 2001.

[TZ05] Mikkel Thorup and Uri Zwick. Approximate distance oracles. *Journal of the ACM*, 52(1):1–24, January 2005.

[TZ06] Mikkel Thorup and Uri Zwick. Spanners and emulators with sublinear distance errors. In 17^{th} *Symposium on Discrete Algorithms (SODA)*, pages 802–809. ACM-SIAM, January 2006.

[Wen91] Rephael Wenger. Extremal graphs with no C^4's, C^6's, or C^{10}'s. *Journal of Combinatorial Theory, Series B*, 52(1):113–116, 1991.

[Woo06] David P. Woodruff. Lower bounds for additive spanners, emulators, and more. In 47^{th} *Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 389–398. IEEE Computer Society Press, October 2006.