Available online at www.sciencedirect.com
ScienceDirect
Theoretical Computer Science 399 (2008) 83–100
Theoretical Computer Science
www.elsevier.com/locate/tcs

Fast deterministic distributed algorithms for sparse spanners

Bilel Derbel*, Cyril Gavoille

Laboratoire Bordelais de Recherche en Informatique, Université de Bordeaux, 351 Cours de la Libération, 33405 Talence, France

Abstract

This paper concerns the efficient construction of sparse and low stretch spanners for unweighted arbitrary graphs with n nodes. All previous deterministic distributed algorithms, for constant stretch spanners of $o(n^2)$ edges, have a running time $\Omega(n^\epsilon)$ for some constant $\epsilon > 0$ depending on the stretch. Our deterministic distributed algorithms construct constant stretch spanners of $o(n^2)$ edges in $o(n^\epsilon)$ time for any constant $\epsilon > 0$.

More precisely, in Linial's free model a.k.a $\mathcal{LOCAL}$ model, we construct in $n^{O(1/\sqrt{\log n})}$ time, for every graph, a $(3, 2)$-spanner of $O(n^{3/2})$ edges, i.e., a spanning subgraph in which the distance is at most 3 times the distance of the original graph plus 2. The result is extended to (α_k, β_k)-spanners with $O(n^{1+1/k} \log k)$ edges for every integer parameter $k \geq 1$, where $\alpha_k + \beta_k = O(k^{\log_2 5})$. If the minimum degree of the graph is $\Omega(\sqrt{n})$, then, in the same time complexity, a $(5, 4)$-spanner with $O(n)$ edges can be constructed.

Keywords: Distributed algorithms; Graph spanners; Time complexity; Linial's free model

1. Introduction

1.1. Motivations

This paper deals with deterministic distributed construction of sparse and low stretch graph spanners. Intuitively, spanners can be thought of as a generalization of the concept of spanning trees. In fact, we look for a spanning subgraph such that the distance between any two nodes in the subgraph is bounded by some constant times the distance in the whole graph. More formally, H is an (α, β)-*spanner* of a graph G if H is a spanning subgraph of G, and if $d_H(u, v) \leqslant \alpha \cdot d_G(u, v) + \beta$ for all nodes u, v of G, where $d_X(u, v)$ denotes the distance from u to v in the graph X. A pair (α, β) for which H is an (α, β)-spanner is called *stretch* of H, and the *size* of H is the number of its edges. An α-spanner is short for an $(\alpha, 0)$-spanner. The quality of a spanner refers to the trade-off between the stretch and the size of the spanner.

The distributed model of computation we will be concerned with is Linial's free model [27], a.k.a. $\mathcal{LOCAL}$ model in [35]. In this model, communication is completely synchronous and reliable. At every time unit, each node may send or receive a message of unlimited size to or from all its neighbors, and can locally compute any function. The model also assumes that each node is equipped with a unique identifier. Much as PRAM algorithms in parallel computing give a good indication of parallelism, the free model gives a good indication of the locality and distributed time.

* Corresponding address: LIFL, Université des Sciences et Technologies de Lille, 59655 Villeneuve d'Ascq Cedex, France.
E-mail addresses: bilel.derbel@lifl.fr (B. Derbel), gavoille@labri.fr (C. Gavoille).

0304-3975/$ - see front matter
doi:10.1016/j.tcs.2008.02.019

From a theoretical point of view, we are interested in the *locality nature* of constructing graph spanners, i.e., what spanners can we compute assuming only some local knowledge? The locality of a distributed problem is often expressed in term of the time needed to resolve it. In fact, in the distributed setting, the best a node can do in $O(t)$ time units is to collect its t neighborhood. For instance, $\Theta(\log^* n)$ time are necessary and sufficient to compute a maximal independent set for trees, bounded degree graphs, or bounded growth graphs with n nodes [11,22,28,23]. Results are known for other fundamental problems such as non-uniform coloring [2,34], minimum spanning tree [17,18,30,29,36], small dominating set [26,39], and maximal matching [24,28].

Graph spanners are the basis of various applications in distributed systems. For instance, Peleg and Ullman [37] establish the relationship between the quality of spanners, and the time and message complexity of network synchronizers (see also [1,33]). Spanners are also implicitly used for the design of low stretch routing schemes with compact tables [12,15,38,40,42], and appear in many parallel and distributed algorithms for computing approximate shortest paths and for the design of compact data structures, a.k.a. distance oracles [9,21,41,43,10].

1.2. Related works

Sparse and low stretch spanners can be constructed from (d, c)-decomposition of Awerbuch and Peleg [6], that is a partition of the graph into clusters of diameter at most d such that the graph obtained by contracting each cluster can be properly c-colored. There are several deterministic algorithms for constructing (d, c)-decompositions [3–5,34]. The resulting distributed algorithms provide $O(k)$-spanners of size $O(n^{1+1/k})$, for any integral parameter $k \geqslant 1$. However, these algorithms run in $\Omega(n^{1/k+\epsilon})$ time, where $\epsilon = \Omega(1/\sqrt{\log n})$, and provide a stretch at least $4k - 3$.

Better stretch-size trade-offs exist but with an increasing time complexity. Recently, a deterministic distributed algorithm has been proposed for constructing a $(2k - 1)$-spanner of size $O(n^{1+1/k})$ in $O(n^{1-1/k})$ time [14]. In particular, 3-spanners of size $O(n^{3/2})$ can be deterministically constructed in $O(\sqrt{n})$ time.

Elkin et al. [16,20,19] develop a distributed algorithm for $(1 + \epsilon, \beta)$-spanners. The size is $O(\beta n^{1+\delta})$ whereas the time is $O(n^{\delta})$, where $\beta = \beta(\delta, \epsilon)$ is independent of n but grows super-polynomially in δ^{-1} and ϵ^{-1}.

Randomized algorithms achieving better performances exist. Baswana et al. [8,7] gave a randomized algorithm which computes respectively a $(2k - 1)$-spanner and a $(k, k - 1)$-spanner with expected size $O(n^{1+1/k})$ in $O(k)$ time. The latter stretch-size trade-off is optimal since, according to an Erdös Conjecture verified for $k = 1, 2, 3, 5$ [44], there are graphs with $\Omega(n^{1+1/k})$ edges and girth $2k + 2$ (the length of the smallest induced cycle), thus for which every (α, β)-spanner requires $\Omega(n^{1+1/k})$ edges if $\alpha + \beta < 2k + 1$. However, as mentioned in [4], a randomized solution might not be acceptable in some cases, especially for distributed computing applications. In the case of graph spanners, deterministic algorithms that *guarantee* a high quality spanner are more than of a theoretical interest. Indeed, one cannot just run a randomized distributed algorithm several times to guarantee a good decomposition, since it is impossible to efficiently check the global quality of the spanner in the distributed model.

1.3. Main results

We consider unweighted undirected connected graphs with n nodes. All previous deterministic distributed algorithms for $O(1)$-spanners of size $o(n^2)$ have a running time $\Omega(n^{\epsilon})$ for some constant $\epsilon > 0$ depending on the stretch. In this paper, we construct constant stretch spanners of size $o(n^2)$ in $o(n^{\epsilon})$ time for any constant $\epsilon > 0$.

More precisely, in the $\mathcal{LOCAL}$ model we construct in $n^{O(1/\sqrt{\log n})}$ time and for every graph a $(3, 2)$-spanner of $O(n^{3/2})$ edges. The result is extended to larger stretch spanners of size $O(n^{1+1/k} \log k)$ for every $k \geqslant 1$. More precisely, we obtain stretches $(\alpha(k), \beta(k))$ which surprisingly depend on the positions of the first two leading 1's in the binary representation of k. A detailed analysis is made for any parameter k and we show that $\alpha(k) + \beta(k)$ is essentially bounded by $k^{\log_2 5}$. Furthermore, for all nodes u and v, the stretch bound on $d_G(u, v)$ depends on whether $d_G(u, v)$ is even or odd. (See the table below. A wider table is available in Section 4.1).

k	1	2	3	4	5
$d_G(u, v) = 1$	$(1, 0)$	$(3, 2)$	$(7, 2)$	$(13, 12)$	$(25, 8)$
$d_G(u, v) > 1$ is even	$(1, 0)$	$(3, 0)$	$(7, 0)$	$(13, 0)$	$(25, 0)$
$d_G(u, v) > 1$ is odd	$(1, 0)$	$(3, 2)$	$(7, -2)$	$(13, 12)$	$(25, -8)$

We also show that if the minimum degree of the graph is $\Omega(\sqrt{n})$, then, in the same time complexity, a $(5, 4)$-spanner with $O(n)$ edges can be constructed.

Our deterministic algorithms have simple randomized versions with improved performances, i.e., $O(\log n)$ time complexity. In particular, we can compute a $(3, 2)$-spanner of size $O(n \log^2 n)$ in $O(\log n)$ time if the minimum degree is $\Omega(\sqrt{n})$.

1.4. Outline of the paper

The main idea to break the $O(n^{1/k+\epsilon})$ time barrier is to abandon the optimality on the stretch-size trade-off. We show that constant stretch spanners can be constructed on the basis of a maximal independent set, i.e., a set of pairwise non-adjacent nodes, maximal for inclusion. This can be deterministically computed in $n^{O(1/\sqrt{\log n})}$ time [4,34]. The time complexity to construct low stretch spanners is improved by a factor of $n^{1/k}$.

The generic algorithm is described in Section 2 and analyzed in Section 3, where a distributed implementation is presented.

We reduce the problem to the computation of an *independent ρ-dominating set*, that is a set X of pairwise non-adjacent nodes such that every node of the graph is at distance at most ρ from X. Using the terminology of [35], an independent ρ-dominating set if nothing else than a (ρ, s)-ruling set[1] for some $s > 1$. Actually, in order to optimize the stretch, our algorithm combines two strategies in a way depending on the binary representation of k.

In Section 4, we present the main results about constant stretch spanners for general graphs. Observing that for $\rho = 1$ an independent ρ-dominating set is a maximal independent set, we conclude that our generic algorithm can be implemented to run in $n^{O(1/\sqrt{\log n})}$ time for $\rho = 1$. Several optimizations are then proposed including randomization and graphs of large minimum degree.

2. A generic algorithm

In this section, we will describe a generic algorithm for constructing a spanner providing a good stretch-size trade-off to be announced later in Section 3.

2.1. Definitions

Let us consider an unweighted undirected connected graph $G = (V, E)$. Given an integer $t \geqslant 1$, the tth power of G, denoted by G^t, is the graph obtained from G by adding an edge between any two nodes at distance at most t in G. For a set of nodes X, $G[X]$ denotes the subgraph of G induced by X. For $X, Y \subseteq V$, let $d_G(X, Y) = \min\{d_G(x, y) \mid x \in X \text{ and } y \in Y\}$.

We will associate with each node $v \in V$ a *region*, denoted by $R(v)$, that is a set of nodes containing v and inducing a connected subgraph of G. Given $U \subseteq V$, G_U denotes the graph whose node set is U, and there is an edge between u and v in U if $d_G(R(u), R(v)) = 1$. We denote by $R^+(v) = \{u \in V \mid d_G(u, R(v)) \leqslant 1\}$ and by $R^+_U(v) = \{u \in U \mid d_G(R(u), R(v)) \leqslant 1\}$.

With a node v, we will associate a color denoted by $c(v)$. Roughly speaking, the color of a node determines the region that the node belongs to.

The *eccentricity* of a node v in G is defined as $\max_{u \in V}\{d_G(u, v)\}$. For a node $v \in X$, we denote by $\mathrm{BFS}(v, X)$ an arbitrary Breadth First Search spanning tree of $G[X]$ rooted at v. We denote by $\mathrm{IDS}(G, \rho)$ an arbitrary independent ρ-dominating set of G. Finally, we define the integer $\ell(x)$ as follows:

$$\ell(x) = \begin{cases} -1 & \text{if } x \leqslant 0, \\ \lfloor \log_2 x \rfloor & \text{otherwise.} \end{cases}$$

In the remainder of this paper we assume the $\mathcal{LOCAL}$ model of computation as defined in the introduction. We define the time complexity of a distributed algorithm to be the worst-case number of time units from the beginning of the algorithm to its termination.

[1] That is a ρ-dominating set X such that $d_G(u, v) \geqslant s$ for all $u \neq v \in X$.

SPANNER

Input: a graph $G = (V, E)$ with $n = |V|$, and integers $\rho, k \geqslant 1$
Output: a spanner S of G with $O(n^{1+1/k} \log k)$ edges

1. $i_0 := \ell(k) - \ell(k - 2^{\ell(k)})$; $U := V$; $r = 0$; $\forall v \in V$, $R(v) := \{v\}$, and $c(v) := v$
2. **for** $i := 1$ **to** $\ell(k) + 1$ **do: if** $i = i_0$ **then** STRATEGY1 **else** STRATEGY2

Fig. 1. Algorithm SPANNER.

STRATEGY1

1. $L := \left\{v \in U,\ |R^+(v)| \leqslant n^{1/k} \cdot |R(v)|\right\}$
2. $\forall (u, v) \in L \times V$ such that $\exists$ edge e between $R(u)$ and v, $S := S \cup \{e\}$
3. $X := \text{IDS}(G^{2(r+1)}[U \setminus L], \rho)$
4. $\forall z \in V$, if $d_G(z, X) \leqslant (2\rho + 1)r + 2\rho$, then set $c(z)$ to be its closest node of X, breaking ties with identities
5. $\forall v \in X$, $R(v) := \{z \in V \mid c(z) = v\}$
6. $\forall v \in X$, $S := S \cup \text{BFS}(v, R(v))$
7. $U := X$ and $r := (2\rho + 1)r + 2\rho$

STRATEGY2

1. $L := \left\{v \in U,\ |R^+_U(v)| \leqslant n^{1/k} \cdot |R(v)|\right\}$
2. $\forall (u, v) \in L \times U$ such that $\exists$ edge e between $R(u)$ and $R(v)$, $S := S \cup \{e\}$
3. $X := \text{IDS}({G_U}^2[U \setminus L], \rho)$
4. $\forall u \in U$, if $d_{G_U}(u, X) \leqslant 2\rho$, then set $c(u)$ to be its closest node of X in G_U, breaking ties with identities
5. $\forall v \in X$, $R(v) := \{R(u) \mid u \in U, c(u) = v\}$
6. $\forall v \in X$, $S := S \cup \text{BFS}(v, R(v))$
7. $U := X$ and $r := (4\rho + 1)r + 2\rho$

Fig. 2. The two strategies.

2.2. Description of the algorithm

The algorithm constructs an efficient clustering of dense regions in the graph. A high level description of the algorithm, named SPANNER, is given in Fig. 1. Intuitively, i_0 represents the distance between the first two leading 1's in the binary representation of k. Typically $i_0 = |p - q|$ if $k = 2^p + 2^q$.

At each iteration of the main loop, one of the two strategies depicted applies (see Fig. 2). The two strategies are very similar. The idea behind the two strategies is the same: choose some well-selected dense regions and merge them with the other ones in order to form new larger regions. The difference is that the density of a region is computed in a different way. The stretch of the output spanner depends on the way the radius of the regions increases and on the total number of phases of the algorithm, depending on the volume of the regions. And, radius and volume increase very differently.

Both strategies create or delete regions each one represented by a *center*. At the beginning of each phase, the set of the region centers is U. There are two types of regions: the *sparse* regions and the *dense* regions. At a given phase, some of the dense regions are selected and enlarged by including nodes from other neighboring regions. One important observation is that each such enlarged region is connected and mutually disjoint.

On the one hand, in STRATEGY1, a region is dense if its neighborhood is $n^{1/k}$ times greater than its size. Applying only STRATEGY1 allows us to obtain small stretch for small values of k. However, asymptotically, the stretch is exponential in k. On the other hand, in STRATEGY2, a region is dense if the number of its neighboring regions is $n^{1/k}$ times greater than its size which provides an exponential growth of the size of a region. Applying only STRATEGY2 allows us to obtain asymptotically stretches polynomial in k.

Algorithm SPANNER switches from one strategy to another at each phase in order to obtain the smallest possible stretch. A full analysis shows that, by alternating STRATEGY1 and STRATEGY2, the best stretch can be obtained by applying STRATEGY1 only once at a well-chosen phase i_0.

We associate with each region $R(v)$ a node, called *center*, and the set of centers forms U. Initially, each node is the center of the region formed by itself. Each phase $i \in \{1, \ldots, \ell(k) + 1\}$ is decomposed in seven parts we briefly sketch.

In Step 1, we partition the regions in sparse (their centers are denoted by L) and dense regions (denoted by $H = U \setminus L$). In Step 2, a sparse region is connected with some neighboring nodes. This step is crucial in the stretch bound analysis. If STRATEGY1 is applied, then each sparse region is connected with each neighboring node in V. So, $R^+(u)$ is spanned for each $u \in L$. If STRATEGY2 is applied, then each sparse region is connected with every neighboring region. Note that at the beginning of a given phase, every region is spanned by a BFS tree constructed in Step 6 of the previous phase.

The centers in H are then processed at the aim of constructing new regions. The key point of our construction is to efficiently merge *all* the regions whose centers are in H into *more dense*, *connected* and *disjoint* regions. In order to guarantee that the algorithm terminates quickly, the dense regions must grow enough. More precisely, if a dense region $R(v)$ is enlarged it must contain at least its neighborhood $R^+(v)$ when STRATEGY1 is applied or its neighborhood in the graph G_U if STRATEGY2 is applied. It is clear that two regions at distance one or two (in G or in G_U depending on the strategy) cannot grow simultaneously without overlapping. Thus, the difficulty is to elect in an efficient way the centers of regions that are allowed to grow in parallel.

In Step 3, we compute an independent ρ-dominating set X in the graph $G^{2(r+1)}[H]$ if STRATEGY1 is applied (resp. ${G_U}^2[H]$ for STRATEGY2), where r is a radius that grows at each phase. The set X defines the set of center regions allowed to grow in parallel.

In order to guarantee that nodes in non-selected regions in Step 3 (the set $H \setminus X$) will be spanned by the output spanner, we must merge them with nodes in the selected regions. Thus, in Step 4, we define a coloring strategy allowing a correct merge process. In fact, in order to ensure that the new regions are disjoint, we let nodes choose their new region in a consistent manner, i.e., a node chooses to be in the region of the closest node in X breaking ties using identities. If STRATEGY1 is applied then each node chooses by itself its dominator, i.e., its closest center in X. However, once a node u chooses its dominator node v, and in order to ensure that the new formed regions are connected, we include all the nodes in the shortest path between u and v, even those in non-dense region. If STRATEGY2 is applied then, the center of each region chooses a dominating region and merge its whole region with it.

In Step 5, the new regions are formed according to the coloring step 4. Note that as soon as the new region are formed, they are spanned in Step 6. Finally, in Step 7, the set of active centers, U, and the variable r are updated for the next phase.

3. Analysis of the algorithm

For every phase i in both strategies, we denote by L_i (resp. X_i) the set L (resp. X) computed during phase i, i.e., after Steps 1 and 3 of phase i. Similarly, we denote by $c_i(z)$ the color of z assigned during phase i, i.e., after Step 4 of phase i. We denote by U_i the set U at the beginning of phase i, and r_i denotes the value of r at the beginning of phase i. Observe that $U_i = X_{i-1}$ for every $i > 1$. For a node $v \in U_i$, we denote by $R_i(v)$ the region of v at the beginning of phase i.

The stretch analysis, which is the most technical part of the paper, is given in Section 3.2, and the size analysis is given in Section 3.3. For these needs, four important properties are presented in the next Section 3.1.

3.1. Properties

Lemma 1. *For every phase i, and for every $v \in U_i$, v is of eccentricity at most r_i in $G[R_i(v)]$.*

Proof. We prove the lemma by induction. The lemma is clearly true for $i = 1$. Let us consider a node $v \in U_i$ and a node $z \in R_i(v)$ at a given phase $i > 1$. We have $U_i = X_{i-1}$. Thus, the region $R_i(v)$ was computed in Step 5 of phase $i - 1$.

- *Case* 1: STRATEGY1 is used at phase $i - 1$. Hence, using Step 5, for every node $z \in R_i(v)$, z was colored v at phase $i - 1$. Thus, $\forall z \in R_i(v), c_{i-1}(z) = v$. Thus, from the coloring step, there exists a shortest path $P = (z = z_1, z_2, z_3, \ldots, z_l, v)$ connecting z and v in G, with $l \leqslant 2(\rho + 1)r_{i-1} + 2\rho$. Let us consider $z_j \in P$ with $1 \leqslant j \leqslant l$. Let us first note that $d_G(z_j, v) \leqslant 2(\rho + 1)r_{i-1} + 2\rho$ and $v \in X_{i-1}$. Hence, z_j has also chosen a color at phase $i - 1$. Using the fact that the coloring is consistent, $c_{i-1}(z_j) = v$. Thus, using Step 5, $P \subseteq R_i(v)$. Hence, $d_{G[R_i(v)]}(z, v) \leqslant 2(\rho + 1)r_{i-1} + 2\rho = r_i$.

– *Case* 2: STRATEGY2 is used at phase $i-1$. Hence, using Step 5, there exists $u \in U_{i-1}$ such that $c_{i-1}(u) = v$ and $z \in R_{i-1}(u)$. Using Step 4, there exists a path P in $G_{U_{i-1}}$ such that $P = (u = u_1, u_2, \ldots, u_l = v)$ with $l \leqslant 2\rho$. From now, it is convenient for the proof of the lemma to view the path P as a directed path where u_1 is the leftmost node and u_l is the rightmost one. From the definition of the graph $G_{U_{i-1}}$, for every pair of successive nodes u_j and u_{j+1}, there exists a pair of neighboring nodes z_j and z_{j+1} such that $z_j \in R_{i-1}(u_j)$, $z_{j+1} \in R_{i-1}(u_{j+1})$ and $(z_j, z_{j+1}) \in E$. Thus, $d_{R_{i-1}(u_j)\cup R_{i-1}(u_{j+1})}(u_j, u_{j+1}) \leqslant d_{R_{i-1}(u_j)}(u_j, z_j) + 1 + d_{R_{i-1}(u_{j+1})}(u_{j+1}, z_{j+1})$. Thus, using the induction hypothesis, $d_{R_{i-1}(u_j)\cup R_{i-1}(u_{j+1})}(u_j, u_{j+1}) \leqslant 2r_{i-1} + 1$. Therefore, $d_{\cup_{1\leqslant j\leqslant l} R_{i-1}(u_j)}(u_1, u_l) \leqslant 2\rho\cdot(2r_{i-1}+1) = 4\rho\cdot r_{i-1}+2\rho$. As we have $z \in R_{i-1}(u = u_1)$, then $d_{\cup_{1\leqslant j\leqslant l} R_{i-1}(u_j)}(z, u_l) \leqslant 4\rho\cdot r_{i-1}+2\rho+r_{i-1} = (4\rho+1)r_{i-1} + 2\rho$.

Notice that $\forall 1 \leqslant j \leqslant l$, $d_{G_{U_{i-1}}}(u_j, v) \leqslant 2\rho$. Hence, u_j has also chosen a color in Step 4 of phase $i-1$. Using the fact that the coloring is consistent, $c_{i-1}(u_j) = v$. Thus, $\forall 1 \leqslant j \leqslant l$, $R_{i-1}(u_j) \subset R_i(v)$. Therefore, $d_{R(v)}(z, v) \leqslant (4\rho+1)r_{i-1} + 2\rho$. By Step 7, we have $r_i = (4\rho+1)r_{i-1} + 2\rho$ which completes the proof. □

Lemma 2. *For every phase i, and for every two nodes $u \neq v \in U_i$, $R_i(u) \cap R_i(v) = \varnothing$.*

Proof. We prove the lemma by induction. The lemma is clearly true for $i = 1$. Let us consider a phase $i > 1$, so $U_i = X_{i-1}$. Let us consider two nodes $u \neq v \in U_i = X_{i_1}$.

– *Case* 1: STRATEGY1 is used at phase $i-1$. Suppose that there exists $z \in R_i(u) \cap R_i(v)$. From Step 5 of phase $i-1$, we have: $c_{i-1}(z) = u$ and $c_{i-1}(z) = v$. Thus, $u = v$ which is a contradiction.
– *Case* 2: STRATEGY2 is used at phase $i-1$. Suppose that there exists $z \in R_i(u) \cap R_i(v)$. From Step 5 of phase $i-1$, there exist $w_1, w_2 \in U_{i-1}$ such that $c_{i-1}(w_1) = u$ and $c_{i-1}(w_2) = v$ and $z \in R_{i-1}(w_1) \cap R_{i-1}(w_2)$. Using the induction hypothesis, $w_1 = w_2$. Thus, $c_{i-1}(w_1) = c_{i-1}(w_2) = u = v$ which is a contradiction.

Therefore for every two nodes $u, v \in U_i$ such that $u \neq v$, $R_i(u) \cap R_i(v) = \varnothing$, which completes the proof. □

Lemma 3. *For every phase $i \neq i_0$, if $|R_i(v)| \geqslant \mathcal{V}$ for every $v \in U_i$, then $|R_{i+1}(w)| \geqslant n^{1/k} \cdot \mathcal{V}^2$ for every $w \in U_{i+1}$.*

Proof. First, because $i \neq i_0$, STRATEGY2 is applied at phase i. Note that $U_{i+1} = X_i$. Let $w \in U_{i+1}$ and $u \in R^+_{U_i}(w)$. Clearly, $u \in U_i \setminus X_i$, otherwise the independence of set X_i is violated. Suppose that there exists a node v' at distance 1 from u in the graph G_{U_i}. Thus, v' is at distance 1 from w in ${G_{U_i}}^2$. Thus, $v' \in U_{i-1} \setminus X_{i-1}$, otherwise the independence of set X_i is violated. Therefore, w is the closest node in X_i to u. Thus, $c_i(u) = w$ and hence, $R_i(u) \subseteq R_{i+1}(w)$. Therefore, by disjointness of the regions (Lemma 2), $|R_{i+1}(w)| \geqslant |R^+_{U_i}(w)| \cdot \mathcal{V}$, where $\mathcal{V} = \min\{|R_i(u)|, u \in U_i \text{ and } c_i(u) = w\}$ because $|R^+_{U_i}(w)|$ is the radius-1 ball of $R_i(w)$ in G_{U_i}. From Step 1 of phase i, $|R^+_{U_i}(v)| > n^{1/k} \cdot |R_i(v)|$ for every $v \in U_{i+1}$, and therefore, for every $v \in U_{i+1}$, $|R_{i+1}(v)| > n^{1/k} \cdot \mathcal{V}^2$. □

Lemma 4. *For every node $u \in V$, there exists a phase i and a node v such that $u \in R_i(v)$, and $v \in L_i \cap U_i$.*

Proof. Let us denote by $i_1 = \ell(k)$ and $i_2 = \ell(k - 2^{\ell(k)})$, i.e., $i_0 = i_1 - i_2$. Let $\mathcal{V}_i$ be the minimum size of any region of a node in U_i.

First, let us show that

Claim 5. $U_{\ell(k)+1} = L_{\ell(k)+1}$.

Proof. Note that $L_i \subseteq U_i$ for every i, so let us show that $U_{\ell(k)+1} \subseteq L_{\ell(k)+1}$.

– *Case* 1: $i_2 = -1$. Hence, $i_0 = \ell(k) + 1$ and $k = 2^{\ell(k)}$. By induction and using Lemma 3, at the beginning of phase i_0, the size of the region of any node in U_{i_0} is at least $\mathcal{V}_{i_0} \geqslant n^{(2^{i_0-1}-1)/k} = n^{(k-1)/k}$ (because $\mathcal{V}_1 = 1$). Hence, for every $v \in U_{i_0}$,

$$n^{1/k} \cdot |R_{i_0}(v)| \geqslant n^{1/k} \cdot n^{(k-1)/k} = n \geqslant |R^+_{i_0}(v)|.$$

As STRATEGY1 is applied at phase i_0, after Step 1, $U_{\ell(k)+1} = L_{\ell(k)+1}$.
– *Case* 2: $i_2 \geqslant 0$. We have $\mathcal{V}_{i_0} \geqslant n^{(2^{i_0-1}-1)/k}$. At phase i_0, we apply STRATEGY1. Thus, using the same arguments than that in Lemma 3, the new enlarged regions constructed at phase i_0 contain all their neighborhood (in G) otherwise the independence of set X_{i_0} is violated. Thus, $\mathcal{V}_{i_0+1} \geqslant n^{1/k} \cdot \mathcal{V}_{i_0} \geqslant n^{2^{i_0-1}/k}$.

· *Subcase* 2.1: $i_2 = 0$. Then, $i_0 = i_1 = \ell(k)$ and $k = 2^{\ell(k)} + 1$. Thus,
$$n^{1/k} \cdot \mathcal{V}^2_{\ell(k)+1} \geqslant n^{(1+2^{\ell(k)})/k} = n.$$
· *Subcase* 2.2: $i_2 \neq 0$. Then, by applying Lemma 3 to phase i going from $i_0 + 1$ to $\ell(k) + 1$ (excluding $\ell(k) + 1$), we have
$$\begin{aligned}\mathcal{V}_{\ell(k)+1} &\geqslant n^{1/k} \cdot \mathcal{V}^2_{\ell(k)} \geqslant n^{1/k} \cdot \left(n^{1/k} \cdot \mathcal{V}^2_{\ell(k)-1}\right)^2 \geqslant \cdots \geqslant n^{(1+2^1+\cdots+2^{j-1})/k} \cdot \mathcal{V}^{2^j}_{\ell(k)+1-j} \\ &\geqslant n^{\sum_{j=0}^{(\ell(k)+1)-(i_0+1)-1} 2^j/k} \cdot \mathcal{V}^{2^{(\ell(k)+1)-(i_0+1)}}_{i_0+1} \geqslant n^{(2^{\ell(k)-i_0}-1+2^{\ell(k)-1})/k}.\end{aligned}$$
Thus,
$$n^{1/k} \cdot \mathcal{V}^2_{\ell(k)+1} \geqslant n^{(1+2(2^{\ell(k)-i_0}-1+2^{\ell(k)-1}))/k} \geqslant n^{(2^{\ell(k)-i_0+1}-1+2^{\ell(k)})/k}.$$
Note that $\ell(k) - i_0 + 1 = i_1 - i_0 + 1 = i_2 + 1$. Thus,
$$2^{\ell(k)-i_0+1} - 1 + 2^{\ell(k)} = 2^{i_2+1} + 2^{i_1} - 1 \geqslant k.$$
Thus, we get $n^{1/k} \cdot \mathcal{V}^2_{\ell(k)+1} \geqslant n$.
Hence, in both subcases, we have
$$\frac{n}{\mathcal{V}_{\ell(k)+1}} \leqslant n^{1/k} \cdot \mathcal{V}_{\ell(k)+1}.$$

Now, let us take a node $v \in U_{\ell(k)+1}$. Because STRATEGY2 is applied at phase $\ell(k) + 1$, v is in $L_{\ell(k)+1}$ iff $|R^+_{U_{\ell(k)+1}}(v)| \leqslant n^{1/k} \cdot |R_{\ell(k)+1}(v)|$. Let us show that, in fact v is in $L_{\ell(k)+1}$. There is at least $\mathcal{V}_{\ell(k)+1}$ nodes in a region, thus using Lemma 2, we have
$$|R^+_{U_{\ell(k)+1}}(v)| \leqslant \frac{n}{\mathcal{V}_{\ell(k)+1}} \leqslant n^{1/k} \cdot \mathcal{V}_{\ell(k)+1} \leqslant n^{1/k} \cdot |R_{\ell(k)+1}(v)|.$$
Thus, all nodes of $U_{\ell(k)+1}$ are in $L_{\ell(k)+1}$.

Therefore, in both cases, $U_{\ell(k)+1} = L_{\ell(k)+1}$, completing the proof of the claim. □

We are now ready to prove Lemma 4.

Let us consider a node $u \in V$. At the beginning of the algorithm, the node u is also in U. If $R(u)$ does not satisfy the condition of Step 1 of the algorithm then $u \in L_1$. Hence, the lemma is true. Otherwise, u participates in Step 3 and u is at distance at most 2ρ from a node in X_1. Thus, u joins the region of some other node $u_1 \in X_1 \subset U_2$ (possibly equal to u) and $u \in R_2(u_1)$. Let us call u_1 the new dominator of u. At the next phase $i = 2$, if u_1 is in L_2 then the lemma holds. Otherwise, u_1 participates in Step 3 and by Step 4, u will be in the region of a new dominator $u_2 \in U_3$. By induction, one can show that, if the $(i-1)$th dominator of u is still not in L_i at phase i, then u will join the region of a new dominator u_i.

In the worst-case, the $\ell(k)$th dominator of u will be in $L_{\ell(k)+1}$ in phase $\ell(k) + 1$. Thus, there must exist some node v such that u is in the region of v at the beginning of some phase i, i.e., v is the $(i-1)$th dominator of u, and v is in the set L_i computed in Step 1 of phase i. □

3.2. Stretch analysis

In the following, we denote $i_1 = \ell(k)$ and $i_2 = \ell(k - 2^{\ell(k)})$, i.e., $i_0 = i_1 - i_2$.

Lemma 6. *For every integer $\rho \geqslant 1$, and for every phase i, we have:*
$$r_i = \begin{cases} (4\rho+1)^{i-1}/2 - 1/2 & \text{if } i \leqslant i_0 \\ (2\rho+1)(4\rho+1)^{i_0-1}/2 + \rho - 1/2 & \text{if } i = i_0 + 1 \\ (2\rho+1)(4\rho+1)^{i-2}/2 + \rho(4\rho+1)^{i-i_0-1} - 1/2 & \text{if } i > i_0 + 1. \end{cases}$$

Proof. Note that $r_1 = 0$, so the result holds for $i = 1$. Assume $i > 1$. STRATEGY2 is applied until phase i_0. So, for every $1 < i \leqslant i_0$, $r_i = (4\rho+1)r_{i-1} + 2\rho$, and by induction we have
$$r_i = (4\rho+1) \cdot \left((4\rho+1)^{i-2}/2 - 1/2\right) + 2\rho = (4\rho+1)^{i-1}/2 - 1/2. \tag{1}$$

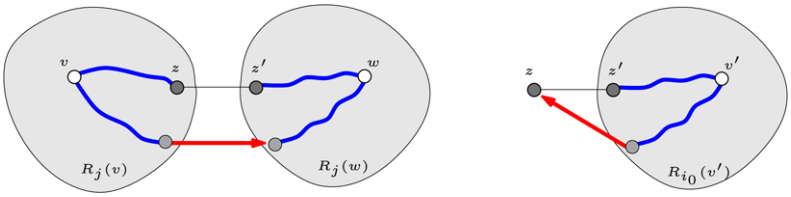

Fig. 3. Stretch analysis for distance 1: *Subcase* 1.1 and *Subcase* 1.2.

In phase i_0, STRATEGY1 is applied. Thus, $r_{i_0+1} = (2\rho+1)r_{i_0} + 2\rho$. Applying Eq. (1) to $i = i_0$, we obtain

$$r_{i_0+1} = (2\rho+1)\left((4\rho+1)^{i_0-1}/2 - 1/2\right) + 2\rho \tag{2}$$

$$= (2\rho+1)(4\rho+1)^{i_0-1}/2 + \rho - 1/2\,. \tag{3}$$

Assume now $i > i_0+1$. STRATEGY2 is applied at each phase $\neq i_0$, so $r_i = (4\rho+1)r_{i-1} + 2\rho$. By induction and by plugging the value of r_{i_0+1} from Eq. (3), we have

$$r_i = (4\rho+1)^{i-(i_0+1)} \cdot r_{i_0+1} + (4\rho+1)^{i-(i_0+1)}/2 - 1/2 \tag{4}$$

$$= (4\rho+1)^{i-(i_0+1)} \cdot \left((2\rho+1)(4\rho+1)^{i_0-1}/2 + \rho - 1/2\right) + (4\rho+1)^{i-(i_0+1)}/2 - 1/2 \tag{5}$$

$$= (2\rho+1)(4\rho+1)^{i-2}/2 + \rho(4\rho+1)^{i-i_0-1} - 1/2. \tag{6}$$

This completes the proof of the lemma. □

Lemma 7. *Let z and z' be two adjacent nodes of G. For all integers $k, \rho \geqslant 1$, the output spanner S of algorithm* SPANNER *satisfies*

$$d_S(z,z') \leqslant \begin{cases} (4\rho+1)^{\ell(k)} & \text{if } k = 2^{\ell(k)} \\ 2(2\rho+1)(4\rho+1)^{\ell(k)-1} + 4\rho(4\rho+1)^{\ell(k-2^{\ell(k)})} - 1 & \text{otherwise.} \end{cases}$$

Proof. Using Lemma 4, there exist a phase j and a node v such that $z \in R_j(v)$ and $v \in U_j \cap L_j$. Similarly for z', there exist j' and v' such that $z' \in R_{j'}(v)$ and $v \in U_{j'} \cap L_{j'}$. W.l.o.g. assume that j and j' are minimum, and that $j \leqslant j'$.

- *Case* 1: $k = 2^{\ell(k)}$, i.e., $i_2 = -1$ and $i_0 = \ell(k)+1$ and. By induction and using Lemma 3, at the beginning of phase i_0, the size of the region of any node in U_{i_0} is at least $n^{(2^{i_0-1}-1)/k} = n^{(k-1)/k}$. Note that we apply STRATEGY1 at phase i_0. Thus, every node in U_{i_0} is in L_{i_0}.
  - *Subcase* 1.1: $j \leqslant j' < i_0$ (see Fig. 3, left-side). Thus, using Step 6, a BFS tree spanning $R_j(v)$ is added to the output spanner at phase $j-1$. In addition, one can easily show that there exists a node $w \in U_j$ such that $z' \in R_j(w)$. Hence, a BFS tree spanning $R_j(w)$ is added to the output spanner at phase $j-1$. Using Step 2 of STRATEGY2, there exists an edge $e \in S$ connecting $R_j(v)$ and $R_j(w)$. Thus, $d_S(z,z') \leqslant 4r_j + 1$. Using Lemma 6, $d_S(z,z') \leqslant 2(4\rho+1)^{j-1} - 1$.
  - *Subcase* 1.2: $j \leqslant j' = i_0$ (see Fig. 3, right-side). In this subcase, STRATEGY1 is applied at phase j'. Using Step 2, $R^+_{j'}(v')$ is spanned. In addition, by Step 6, a BFS tree spanning $R_{j'}(v')$ is added to the output spanner at phase $j'-1$. Thus, because $z \in R^+_{j'}(v')$, $d_S(z,z') \leqslant 2r_{j'} + 1 = 2r_{i_0} + 1$. Using Lemma 6, $d_S(z,z') \leqslant (4\rho+1)^{i_0-1}$.

  Finally, because $\rho > 0$, in both subcases, the stretch is bounded by $(4\rho+1)^{\ell(k)}$.
- *Case* 2: $k \neq 2^{\ell(k)}$, i.e., $i_2 \geqslant 0$.
  - *Subcase* 2.1: $j \neq i_0$. Thus, it easy to show that there exists a node $w \in U_j$ such that $z' \in R_j(w)$. Using Step 6, $R_j(w)$ and $R_j(v)$ were spanned by a BFS tree at phase $j-1$. In addition, because STRATEGY2 is applied at phase j, an edge e connecting $R_j(v)$ and $R_j(w)$ is added at phase j (Step 2). Thus, $d_S(z,z') \leqslant 4r_j + 1$.
  - *Subcase* 2.2: $j = i_0$. Thus, STRATEGY1 is applied at phase j and $R^+_j(v)$ is spanned by a BFS tree. Since $z' \in R^+_j(v)$, we have $d_S(z,z') \leqslant 2r_j + 1 = 2r_{i_0} + 1$.

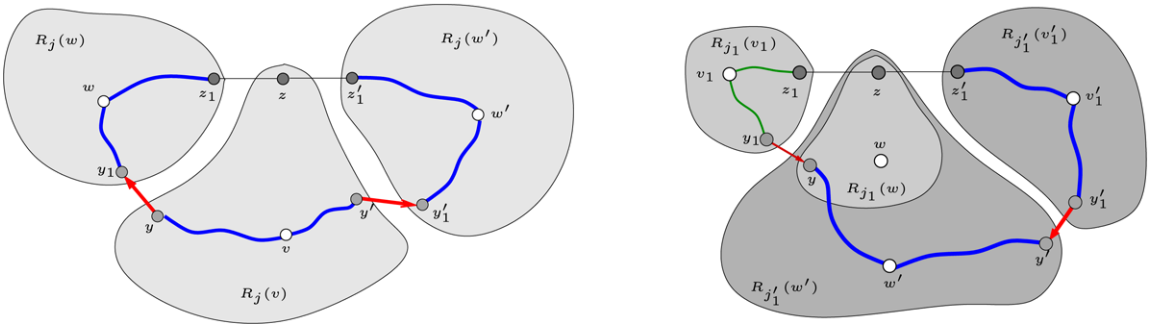

Fig. 4. Stretch analysis for distance 2: *Subcase* 1.1 and *Subcase* 1.2.

Thus, the stretch is bounded by $4r_{\ell(k)+1}+1$. Using Lemma 6, we have:

$$\begin{aligned} 4r_{\ell(k)+1}+1 &= 4\left((4\rho+1)^{i_2}\cdot r_{i_0+1}+\frac{1}{2}((4\rho+1)^{i_2}-1)\right)+1 \\ &= 2(2\rho+1)(4\rho+1)^{i_1-1}+4\rho(4\rho+1)^{i_2}-1. \quad \square \end{aligned}$$

From Lemma 7, we can obtain a general bound for the stretch of the output spanner S of algorithm SPANNER. However, in the next lemmas, by considering distance 2 and distance 3 nodes, we give a different analysis which provides improved bounds.

Lemma 8. *Let z_1 and z'_1 (resp. z_2 and z'_2) be two nodes at distance 2 (resp. at distance 3). For all integers $k, \rho \geqslant 1$, if $i_0 = \ell(k)+1$, i.e., $k = 2^{\ell(k)}$, then the output spanner S of algorithm* SPANNER *satisfies:*

$$\begin{cases} d_S(z_1, z'_1) \leqslant (4\rho+1)^{\ell(k)}+1 \\ d_S(z_2, z'_2) \leqslant 2(4\rho+1)^{\ell(k)}+1. \end{cases}$$

Proof. In the following proof, we shall keep in mind that in the case $k = 2^{\ell(k)}$, STRATEGY1 is applied in the last phase $\ell(k)+1 = i_0$.

First let us study the stretch for the two nodes z_1 and z'_1 satisfying $d_G(z_1, z'_1) = 2$. Let us consider a path (z_1, z, z'_1) of length 2 in G. Using Lemma 4, there exist a phase j (resp. j_1 and j'_1) and a node v (resp. v_1 and v'_1) such that $v \in U_j$ (resp. $v_1 \in U_{j_1}$ and $v'_1 \in U_{j'_1}$), $z \in R_j(v)$ (resp. $z_1 \in R_{j_1}(v_1)$ and $z'_1 \in R_{j'_1}(v'_1)$) and $v \in L_j$ (resp. $v_1 \in L_{j_1}$ and $v'_1 \in L_{j'_1}$). W.l.o.g. assume that j, j_1, j'_1 are minimum.

In the following, we analyze of all possible cases.

– *Case* 1: $j \neq i_0$, i.e., z belongs to a sparse region before the last phase $\ell(k)+1 = i_0$.

· *Subcase* 1.1: $j \leqslant j_1$, and $j \leqslant j'_1$ (see Fig. 4, left-side). Let $R_j(w)$ and $R_j(w')$ the regions containing z_1 and z'_1 at phase j. Since the regions $R_j(v)$, $R_j(w)$ and $R_j(w')$ are neighbors and using Step 2 of STRATEGY2, there exist nodes y, y_1, y' and y'_1 (respectively in regions $R_j(v)$, $R_j(w)$, $R_j(v)$ and $R_j(w')$) such that $e_1 = (y, y_1)$ is an edge in the spanner S connecting $R_j(v)$ with $R_j(w)$ and $e'_1 = (y', y'_1)$ is an edge in the spanner S connecting $R_j(v)$ with $R_j(w')$. In addition, using Step 6, the regions $R_j(v)$, $R_j(w)$ and $R_j(w')$ were spanned by a BFS tree in phase $j-1$. Thus, we have:

$$\begin{aligned} d_S(z_1, z'_1) &\leqslant d_S(z_1, v_1)+d_S(v_1, y_1)+d_S(y_1, y)+d_S(y, v)+d_S(v, y') \\ &\quad +d_S(y', y'_1)+d_S(y'_1, v'_1)+d_S(v'_1, z'_1) \\ &= r_j+r_j+1+r_j+r_j+1+r_j+r_j \\ &\leqslant 6r_{i_0-1}+2. \end{aligned}$$

· *Subcase* 1.2: $j > j_1$, and $j > j'_1$ (see Fig. 4, right-side). W.l.o.g. assume that $j_1 \leqslant j'_1$. By construction, there exists a node w such that $z \in R_{j_1}(w)$, i.e., $R_{j_1}(w)$ is the region containing z in phase j_1. Since $R_{j_1}(v_1)$ and $R_{j_1}(w)$ are neighbors and using Step 2, there exist nodes $y_1 \in R_{j_1}(v_1)$ and $y \in R_{j_1}(w)$ such that (y, y_1) is an edge in the spanner S connecting $R_{j_1}(w)$ and $R_{j_1}(v_1)$. Thus, using Step 6, we have:

$$d_S(z_1, y) \leqslant 2r_{j_1}+1 \leqslant 2r_{i_0-1}.$$

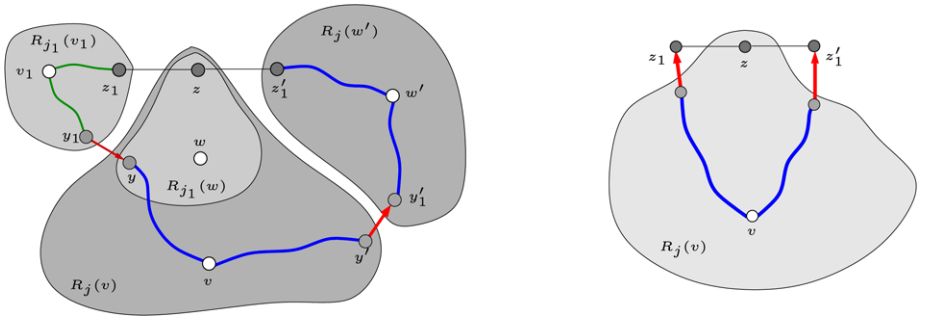

Fig. 5. Stretch analysis for distance 2: *Subcase* 1.3 and *Case* 2.

Let us suppose that $j'_1 > j_1$ and let us focus on the region $R_{j_1}(w)$ containing both z and y. We only apply STRATEGY2 in phases before phase $j'_1 \leqslant j < i_0$. Thus, at each phase where STRATEGY2 is applied, and using Steps 4 and 5, the whole region $R_{j_1}(w)$ is entirely merged with neighboring ones in order to form a new enlarged region. Then, the new enlarged region is entirely merged with other regions and so on. Thus, nodes z and y will always belong to the same region (which is connected). In particular, there exists a region $R_{j'_1}(w')$ such that both z and y are in $R_{j'_1}(w')$. Since $R_{j'_1}(v'_1)$ and $R_{j'_1}(w')$ are neighbors, there exist nodes $y' \in R_{j'_1}(w')$ and $y'_1 \in R_{j'_1}(w')$ such that (y', y'_1) is an edge of the output spanner S connecting $R_{j'_1}(w')$ and $R_{j'_1}(w')$. Thus, $d_S(z'_1, y') \leqslant 2r_{j'_1} + 1 \leqslant 2r_{i_0-1} + 1$. Using Step 6, $R_{j'_1}(w')$ is spanned by a BFS tree. Thus, $d_S(y, y') \leqslant 2r_{j'_1} \leqslant 2r_{i_0-1}$. Thus,

$$d_S(z_1, z'_1) \leqslant 6r_{i_0-1} + 2.$$

If $j'_1 = j_1$, then it is easy to see that the region $R_{j'_1}(v'_1)$ is connected to $R_{j_1}(w)$ using an edge in the output spanner. Hence, it is easy to find a path in S such that $d_S(z_1, z'_1) \leqslant 2r_{j_1} + 1 + 2r_{j_1} + 1 + 2r_{j_1} = 6r_{i_0-1} + 2$.

- *Subcase* 1.3: $j_1 \leqslant j \leqslant j'_1$ (see Fig. 5, left-side) which is symmetric to $j'_1 \leqslant j \leqslant j_1$. Here, the only difference with subcase 1.2 is that the region $R_j(v)$ becomes sparse before $R_{j'_1}(v'_1)$. It is straightforward from the analysis of subcase 1.2 and Fig. 5 (left-side) that: $d_S(z_1, z'_1) \leqslant 6r_{i_0-1} + 2$.

– *Case* 2: $j = i_0$ (see Fig. 5, right-side). Thus, STRATEGY1 is applied at phase j. It is clear by Lemma 4 and the analysis there that $R_j(v) \in L_{j=i_0}$. Thus, the neighborhood $R^+_j(v)$ of $R_j(v)$ is spanned by a BFS tree. Since, $z_1, z'_1 \in R^+_j(v)$, we get: $d_S(z_1, z'_1) \leqslant 2r_{i_0} + 2$.

Thus, in all cases, the following holds: $d_S(z_1, z'_1) \leqslant \max\{2r_{i_0} + 2, 6r_{i_0-1} + 2\}$.
But $r_{i_0} = (4\rho + 1)r_{i_0-1} + 2\rho$. Hence,

$$\begin{aligned} d_S(z_1, z'_1) &\leqslant \max\{2(4\rho + 1)r_{i_0-1} + 4\rho + 2, 6r_{i_0-1} + 2\} \\ &= 2r_{i_0} + 2. \end{aligned}$$

Using Lemma 6, we have $d_S(z_1, z'_1) \leqslant 2 \cdot (\frac{1}{2} \cdot ((4\rho + 1)^{i_0-1} - 1)) + 2 = (4\rho + 1)^{\ell(k)} + 1$. This demonstrates the lemma for the case of distance 2 nodes.

The case of distance 3 nodes z_2 and z'_2 follows easily from Lemma 7. In fact, consider a node z in the shortest path (in G) between z_2 and z'_2. W.l.o.g, suppose that z is at distance 2 (resp. 1) from z_2 (resp. from z'_2) in G. Then, $d_S(z_2, z'_2) \leqslant d_S(z_2, z) + d_S(z, z'_2)$. Hence, using the previous bound for distance 2 nodes and Lemma 7, we have $d_S(z_2, z'_2) \leqslant (4\rho + 1)^{\ell(k)} + 1 + (4\rho + 1)^{\ell(k)}$. This concludes the proof. □

Lemma 9. *Let z_1 and z'_1 (resp. z_2 and z'_2) be two nodes at distance 2 (resp. at distance 3). For all integers $k, \rho \geqslant 1$, if $i_0 < \ell(k) + 1$, then the output spanner S of algorithm* SPANNER *satisfies:*

$$\begin{cases} d_S(z_1, z'_1) \leqslant 3(2\rho + 1)(4\rho + 1)^{\ell(k)-1} + 6\rho(4\rho + 1)^{\ell(k-2^{\ell(k)})} - 1 \\ d_S(z_2, z'_2) \leqslant 4(2\rho + 1)(4\rho + 1)^{\ell(k)-1} + 8\rho(4\rho + 1)^{\ell(k-2^{\ell(k)})} - 1. \end{cases}$$

Proof. First let us study the stretch for the two nodes z_1 and z'_1 which satisfy $d_G(z_1, z'_1) = 2$. Let us consider a path (z_1, z, z'_1) of length 2 in G. Using Lemma 4, there exists a phase j (resp. j_1 and j'_1) and a node v (resp. v_1 and v'_1)

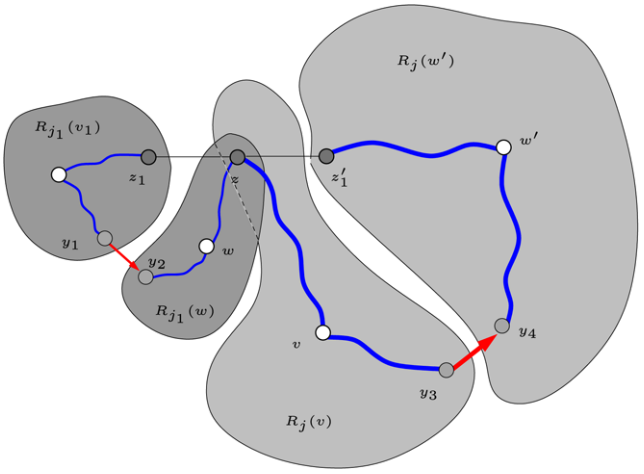

Fig. 6. Stretch analysis for distance 2: $j_1 < i_0 < j \leqslant j'_1$.

such that $v \in U_j$ (resp. $v_1 \in U_{j_1}$ and $v'_1 \in U_{j'_1}$), $z \in R_j(v)$ (resp. $z_1 \in R_{j_1}(v_1)$ and $z'_1 \in R_{j'_1}(v'_1)$) and $v \in L_j$ (resp. $v_1 \in L_{j_1}$ and $v'_1 \in L_{j'_1}$). W.l.o.g. assume that j, j_1, j'_1 are minimum.

– *Case* 1: $j_1 \leqslant j \leqslant j'_1$. Here the critical subcase is when $j_1 < i_0 < j \leqslant j'_1$ (see Fig. 6).
In fact, we can find two nodes $y_1 \in R_{j_1}(v_1)$ and $y_2 \in R_{j_1}(w)$ where $R_{j_1}(w)$ is the region containing z in phase j_1 such that the edge (y_1, y_2) is in the spanner. Then, the region $R_{j_1}(w)$ is entirely merged with other regions until phase i_0. At phase $i_0 < j$, it may happen that the region $R_{j_1}(w)$ becomes broken into many parts and the nodes y_1 and y_2 become in two new different regions. Thus, it may happen that $R_{j_1}(w) \nsubseteq R_j(v)$. (This is the main difference with the case $k = 2^{\ell(k)}$). Nevertheless, we can find two nodes $y_3 \in R_j(v)$ and $y_4 \in R_j(w')$, where $R_j(w')$ is the region containing z'_1 at phase j, such that the edge (y_3, y_4) is in the output spanner. Thus,

$$d_S(z_1, z'_1) \leqslant 4r_{j_1} + 1 + 4r_j + 1 \leqslant 4r_{i_0-1} + 1 + 4r_{\ell(k)+1} + 1.$$

In the other subcases (depending on the position of phase i_0 compared with phases j_1, j and j'_1), we have essentially the same analysis as for the case $k = 2^{\ell(k)}$ and it is not difficult to show that $d_S(z_1, z'_1) \leqslant 6r_{\ell(k)+1} + 2$. Thus, we obtain:

$$d_S(z_1, z'_1) \leqslant \max\left\{6r_{\ell(k)+1} + 2,\ 4r_{i_0-1} + 4r_{\ell(k)+1} + 2\right\}.$$

– Other cases: By checking the other cases one by one, one can verify that the previous bound is still true.

Using Lemma 6 and by a simple checking, one can show that $\left(6r_{\ell(k)+1} + 2\right) - \left(4r_{i_0-1} + 4r_{\ell(k)+1} + 2\right) \geqslant 0$. Using Lemma 6 to compute $6r_{\ell(k)+1} + 2$ concludes the proof of the lemma for distance 2 nodes.

Now, let us study the stretch for the two nodes z_2 and z'_2 satisfying $d_G(z_2, z'_2) = 2$. Let us consider a path (z_2, z, z', z'_2) of length 3 in G. Using Lemma 4, there exists a phase j (resp. j', j_2 and j'_2) and a node v (resp. v', v_2 and v'_2) such that $v \in U_j$ (resp. $v' \in U_{j'}$, $v_2 \in U_{j_2}$ and $v'_2 \in U_{j'_2}$), $z \in R_j(v)$ (resp. $z \in R_{j'}(v')$, $z_2 \in R_{j_2}(v_1)$ and $z'_2 \in R_{j'_2}(v'_2)$) and $v \in L_j$ (resp. $v' \in L_{j'}$, $v_2 \in L_{j_2}$ and $v'_2 \in L_{j'_2}$). W.l.o.g. assume that j, j', j_2, j'_2 are minimum.

There are too many cases to detail all of them. Thus, we will just give the bound obtained for each case. Using the same ideas than previously, the reader can guess the path of S allowing to obtain the corresponding bound.

– *Case* 1: $j_2 \leqslant j \leqslant j' \leqslant j'_2$. The critical case is when $j_2 \leqslant i_0 \leqslant j$ or $j \leqslant i_0 \leqslant j'$. In fact, we have:
  · If $j_2 < i_0 \leqslant j$ (see Fig. 7 left-side), then $d_S(z_2, z'_2) \leqslant 4r_{j_2} + 1 + 2r_j + 1 + 2r_{j'} + 1 + 2r_{j'}$. Thus, $d_S(z_2, z'_2) \leqslant 4r_{i_0-1} + 6r_{\ell(k)+1} + 3$.
  · If $j < i_0 \leqslant j'$ (see Fig. 7, right-side), then $d_S(z_2, z'_2) \leqslant 2r_{j_2} + 1 + 2r_j + 1 + 2r_j + 2r_{j'} + 1 + 2r_{j'}$. Thus, $d_S(z_2, z'_2) \leqslant 6r_{i_0-1} + 4r_{\ell(k)+1} + 3$.
  · Otherwise, one can easily see that $d_S(z_2, z'_2) \leqslant 8r_{\ell(k)+1} + 3$. For instance, if $j_2 \leqslant j \leqslant j' \leqslant j'_2 < i_0$ (or if $i_0 < j_2 \leqslant j \leqslant j' \leqslant j'_2$), then we have the situation depicted on Fig. 8 (left-side) which is the same that for

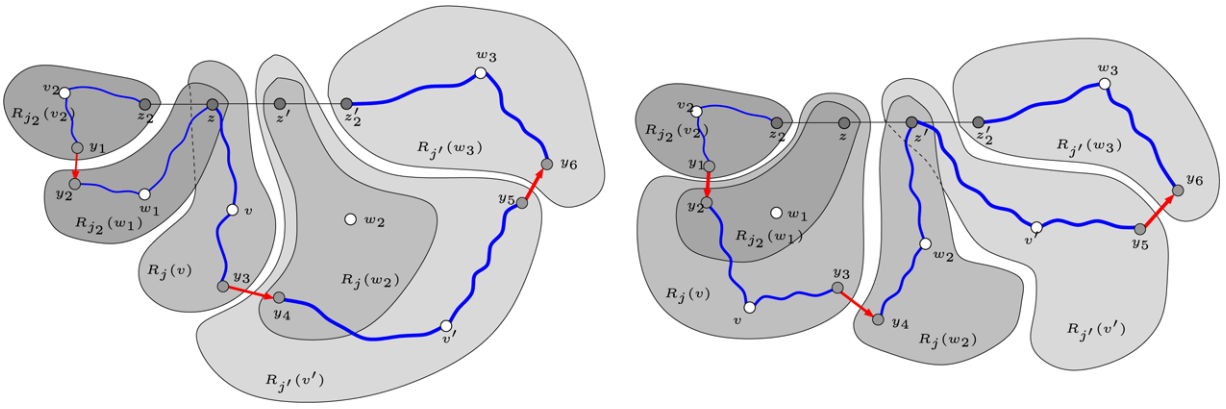

Fig. 7. Stretch analysis for distance 3 (k not power of 2): $j_2 < i_0 \leqslant j \leqslant j' \leqslant j'_2$ (left), and $j_2 \leqslant j < i_0 \leqslant j' \leqslant j'_2$ (right).

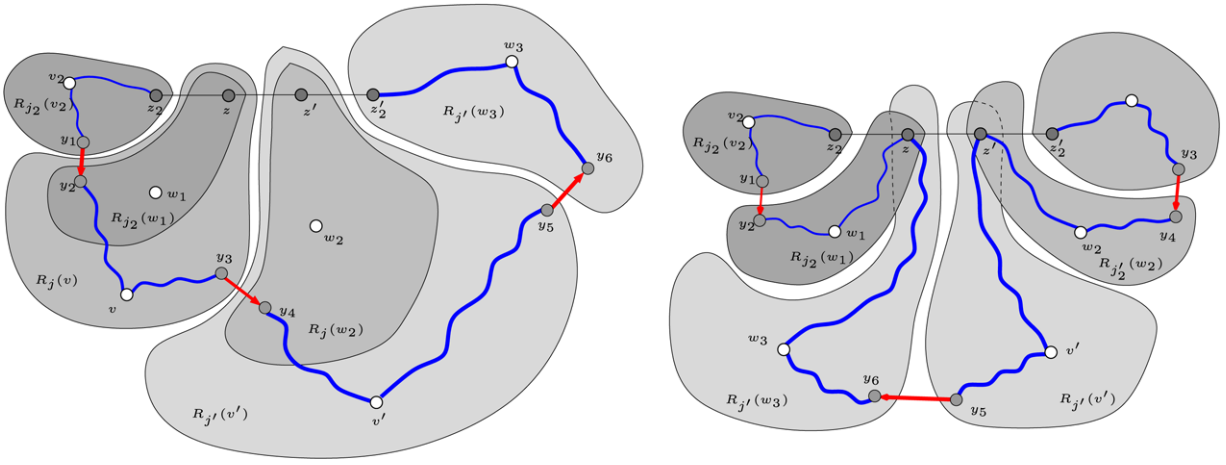

Fig. 8. Stretch analysis for distance 3 (k not power of 2): $j_2 \leqslant j \leqslant j' \leqslant j'_2 < i_0$ (left), and $j_2 \leqslant j'_2 < i_0 \leqslant j' \leqslant j$ (right).

the case k power of 2. All the other subcases are very similar and are based on the observation that if two nodes belong to the same region at phase $i < i_0$ (resp. $i > i_0$), then in any phase i' such that $i < i' \leqslant i_0$ (resp. $i < i'$) the two nodes will still belong to a common region, i.e., a region is never broken by STRATEGY2.

– *Case* 2: $j_2 \leqslant j \leqslant j'_2 \leqslant j'$. Here, we obtain the same bounds as for Case 1.
– *Case* 3: $j_2 \leqslant j'_2 \leqslant j' \leqslant j$. Here, the critical cases are:
  · $j'_2 < i_0$ and we have the situation of Fig. 8 (right-side). Thus, we obtain:
    $$d_S(z_2, z'_2) \leqslant 4r_{j_2} + 1 + 4r_{j'} + 1 + 4r_{j_2} + 1 \leqslant 8r_{i_0-1} + 4r_{\ell(k)+1} + 3$$
  · $j_2 < i_0 < j'_2$ and we obtain $d_S(z_2, z'_2) \leqslant 4r_{i_0-1} + 6r_{\ell(k)+1} + 3$.
– Other cases: By studying all the remaining cases, we always get similar situations than that in the previous cases, and it is not difficult to show that we obtain the same upper bounds.

At final, and in all cases, we have:

$$d_S(z_2, z'_2) \leqslant \max\left\{8r_{\ell(k)+1} + 3, 6r_{i_0-1} + 4r_{\ell(k)+1} + 3, 4r_{i_0-1} + 6r_{\ell(k)+1} + 3, 8r_{i_0-1} + 4r_{\ell(k)+1} + 3\right\}.$$

Thus, by a routine computation based on Lemma 6, one can check that:

$$d_S(z_2, z'_2) \leqslant 8r_{\ell(k)+1} + 3 = 4(2\rho+1)(4\rho+1)^{\ell(k)-1} + 8\rho(4\rho+1)^{\ell(k-2^{\ell(k)})} - 1. \quad \square$$

3.3. Size analysis

Lemma 10. *For all integers k, $\rho \geqslant 1$, the size of the output spanner S of algorithm* SPANNER *is $O(n^{1+1/k} \log k)$.*

Proof. The output spanner S is updated in Steps 2 and 6 of each phase. Let us consider two consecutive phases i and $i-1$ and the edges added by Step 6 at phase $i-1$ and the edges added by Step 2 at phase $i > 1$.

– *Case* 1: STRATEGY1 is applied at phase i. Thus, the number of edges is bounded by:

$$\begin{aligned}\sum_{v\in L_i} |\mathrm{BFS}(v, R_i(v))| + \sum_{v\in L_i} |R^+(v)| &\leqslant n + \sum_{v\in L_i} |R^+(v)| \\ &\leqslant n + \sum_{v\in L_i} n^{1/k} |R_i(v)| \\ &\leqslant n + n^{1+1/k}.\end{aligned}$$

– *Case* 2: STRATEGY2 is applied at phase i. Thus, the number of edges added is bounded by:

$$\begin{aligned}\sum_{v\in L_i} |\mathrm{BFS}(v, R_i(v))| + \sum_{v\in L_i} |R^+_{U_i}(v)| &\leqslant n + n^{1/k} \sum_{v\in L_i} |R_i(v)| \\ &\leqslant n + n^{1+1/k}.\end{aligned}$$

Since there are $O(\log k)$ phases in the algorithm, the lemma is true. □

3.4. Distributed implementation and time complexity

In the $\mathcal{LOCAL}$ model, distributed computation of some distributed procedure A on $G^t[H]$ can be easily simulated on G as follows, charging the overall time by a factor of t. Hereafter, we assume that each node $u \in G$ can determine if it belongs or not to H. Indeed, consider one communication step in A running on some node u of $G^t[H]$ followed by one local computation step. In G, an original message in A is sent from u with a counter initialized to $t-1$ as an extra field. Now, each node $v \in G$, upon the reception of a message with some counter in its header: (1) decrements the counter; (2) stores this message if $v \in H$; and (3) forwards the incoming message with the updated counter to all its neighbors in G if the updated counter is non-null (if many messages are received during a round, then they are concatenated before being sent). After t communication rounds in G, every node $u \in H$ starts the local computation step of A on the base of all received messages during the last t communication rounds.

Similarly, given $U \subseteq V$, the computation of some distributed procedure A on G_U can be simulated on G as follows, charging the overall time by a factor $O(r)$ where r is an upper bound of the eccentricity of a node $v \in U$ in $G[R(v)]$. At each time procedure A requires for a node v of G_U to send a message to a neighbor, v broadcasts the message in $G[R(v)]$ (which is connected). The nodes at the frontier of $R(v)$, i.e., nodes having neighbors in different regions, also broadcasts the message out of their region. Symmetrically, upon the reception of messages from different regions, messages are concatenated and a convergecast is performed to v. The time overhead for one step of A is at most $2r+1$.

Relying on the above discussions, running procedure A on $G^{2(r+1)}[H]$ or on ${G_U}^2[H]$ can be simulated on G within a factor of $O(r)$ on the time complexity.

Lemma 11. *For all integers k, $\rho \geqslant 1$,* SPANNER *can be implemented with a deterministic distributed algorithm in $k^{O(\log \rho)} \cdot \tau$ time, where τ is the time complexity to compute an independent ρ-dominating set in a graph of at most n nodes.*

Proof. Let us first remark that in the $\mathcal{LOCAL}$ model, we can make the nodes know their entire p-neighborhood in $O(p)$ time. Therefore, any task which requires only information about p-neighborhood can be solved within $O(p)$ time.

Let us consider a fixed phase $i \leqslant \ell(k)$. For every $v \in U$, Steps 1 and 6 of the algorithm can be implemented by traversing $R_i^+(v)$ a constant number of times which is $O(r_i)$ time consuming. The time needed to run procedure IDS on $G^{2(r_i+1)}[H]$ blows up by a factor of $2(r_i+1)$ as explained before. Thus, if STRATEGY1 is applied, then Step 3 is

$O(r_i \cdot \tau)$ time consuming. Similarly, the time complexity of procedure IDS on $G_U[H]$ blows up by a factor $2r_i$. Thus, if STRATEGY2 is applied, Step 3 is $O(r_i \cdot \tau)$ time consuming.

Steps 4 and 5 can be implemented by letting each node exploring its $O(2(r_i+1)\rho + r_i)$-neighborhood which is $O(r_i)$ time consuming for fixed ρ. Finally, Step 7 is $O(1)$ time consuming. Summing up among all steps, every phase $i \leqslant \ell(k)$ is $O(r_i \cdot \tau)$ time consuming.

At phase $\ell(k)+1$, the set $U \setminus L$ is empty and thus phase $\ell(k)+1$ is $O(r_{\ell(k)+1})$ time consuming.

Using Lemmas 1 and 6 and summing up among all phases, the time complexity of the algorithm is $O((4\rho+1)^{\ell(k)} \cdot \tau)$. Since $\ell(k) = O(\log k)$, the time complexity is bounded by $\rho^{O(\log k)} \cdot \tau = k^{O(\log \rho)} \cdot \tau$ which completes the proof. □

4. Applications to low stretch spanners

4.1. Constant stretch spanners with subquadratic size

In this section we are interested in small values of k. The stretch is optimized for special values of k that are on the form $k = 2^p + 2^q - 1$ for integers $p, q \geqslant 0$. This include all powers of two ($q = 0$), and we observe that all integers from 1 to 9, but 6, are on this form.

Let MIS(n) denote the time complexity for computing, by a deterministic distributed algorithm, a maximal independent set (MIS) in a graph with at most n nodes. The fastest deterministic algorithm [34] shows that $\text{MIS}(n) \leqslant n^{O(1/\sqrt{\log n})}$. It is also known that $\text{MIS}(n) \geqslant \Omega(\sqrt{\log n / \log\log n})$ [24].

It is not difficult to check that a set X is an independent 1-dominating set if and only if X is a maximal independent set (cf. [35, pp. 259, Ex. 4]). In this part we therefore concentrate our attention on $\rho = 1$. Thus, using the fast distributed MIS algorithm as a subroutine in algorithm SPANNER with $\rho = 1$, we obtain:

Theorem 12. *There is a deterministic distributed algorithm that given a graph G with n nodes and any integer $k = 2^p + 2^q - 1$ with $p \geqslant q \geqslant 0$, constructs a spanner for G with $O(n^{1+1/k} \log k)$ edges in $O(k^{O(1)}\text{MIS}(n))$ time, with the following stretch properties, $\forall u, v \in V$:*

If $q = 0$, i.e., $k = 2^p$, then:

- *If $d_G(u,v)$ is even, then: $d_S(u,v) \leqslant \frac{1}{2}(5^p+1) \cdot d_G(u,v)$.*
- *Otherwise, $d_S(u,v) \leqslant \frac{1}{2}(5^p+1) \cdot d_G(u,v) + \frac{1}{2}(5^p-1)$.*

If $q > 0$, then:

- *If $d_G(u,v) = 1$, then: $d_S(u,v) \leqslant 6 \cdot 5^{p-1} + 4 \cdot 5^{q-1} - 1$.*
- *If $d_G(u,v)$ is even, then: $d_S(u,v) \leqslant \frac{1}{2}\left(9 \cdot 5^{p-1} + 6 \cdot 5^{q-1} - 1\right) \cdot d_G(u,v)$.*
- *Otherwise, $d_S(u,v) \leqslant \frac{1}{2}\left(9 \cdot 5^{p-1} + 6 \cdot 5^{q-1} - 1\right) \cdot d_G(u,v) - \frac{1}{2}\left(3 \cdot 5^{p-1} + 2 \cdot 5^{q-1} - 1\right)$.*

Proof. We just detail the stretch analysis, size and time which are direct consequences of Lemmas 3 and 11 for $\rho = 1$.

First, let us assume that $q = 0$, i.e., $k = 2^p = 2^{\ell(k)}$.

If $d_G(u,v) = 2t$ for some integer $t \geqslant 0$, then we consider a shortest path between u and v in G. This path can be viewed as the sum of t segments of length 2 each. Now, using Lemma 8, the stretch of each segment is $5^p + 1$. Thus $d_S(u,v) \leqslant (5^p+1) \cdot t = \frac{1}{2}(5^p+1) d_G(u,v)$ and the stretch bound for even distances holds.

If $d_G(u,v) = 2t+1$ for some integer $t > 0$, then we consider a shortest path between u and v in G. It can be viewed as the sum of $t-1 \geqslant 0$ segments of length 2, and a segment of length 3. Now, using Lemma 8, the stretch of each even segment is $5^p + 1$ and the stretch of the length 3 segment is $2 \cdot 5^p + 1$. In total, $d_S(u,v) \leqslant (5^p+1)(t-1) + 2 \cdot 5^p + 1 = (5^p+1) \cdot (\frac{1}{2}(d_G(u,v)-1)-1) + 2 \cdot 5^p + 1 = \frac{1}{2}(5^p+1) \cdot d_G(u,v) + \frac{1}{2}(5^p-1)$, and the stretch bound for odd distances holds.

If $d_G(u,v) = 1$, then using Lemma 7, $d_S(u,v) \leqslant 5^p = \frac{1}{2}(5^p+1) \cdot d_G(u,v) + \frac{1}{2}(5^p-1)$.

Assume now that $q > 0$.

First, if $p = q$, then $k = 2^{p+1} - 1 = \sum_{j=0}^{p} 2^j$. Hence, $\ell(k) = p$ and $\ell(k - 2^{\ell(k)}) = \ell(2^{p+1} - 1 - 2^p) = \ell(2^p - 1) = p - 1$. If $p \neq q$, then $k = 2^p + \sum_{j=0}^{q-1} 2^j$. Hence, $\ell(k) = p$. In addition, $\ell(k - 2^{\ell(k)}) = \ell(2^p + 2^q - 1 - 2^p) = \ell(2^q - 1) = q - 1$.

k	1	2	3	4	5	7	8	9
$(p,q):i_0$	$(0,0):1$	$(1,0):2$	$(1,1):1$	$(2,0):3$	$(2,1):2$	$(2,2):1$	$(3,0):4$	$(3,1):3$
$d_G(u,v)=1$	$(1,0)$	$(3,2)$	$(7,2)$	$(13,12)$	$(25,8)$	$(37,12)$	$(63,62)$	$(115,38)$
$d_G(u,v)>1$ is even	$(1,0)$	$(3,0)$	$(7,0)$	$(13,0)$	$(25,0)$	$(37,0)$	$(63,0)$	$(115,0)$
$d_G(u,v)>1$ is odd	$(1,0)$	$(3,2)$	$(7,-2)$	$(13,12)$	$(25,-8)$	$(37,-12)$	$(63,62)$	$(115,-38)$

Fig. 9. Stretches (α_k,β_k) for $k=2^p+2^q-1$.

If $d_G(u,v)=1$, the stretch bound is given by Lemma 7 (with $\rho=1$).

If $d_G(u,v)=2t$ for some $t\geqslant 0$, then by viewing the shortest path between u and v as a sum of segments of length 2 and using Lemma 9, we have $d_S(u,v)\leqslant(9\cdot 5^{p-1}+6\cdot 5^{q-1}-1)t$. Hence, the stretch bound for even distance holds.

Otherwise, if $d_G(u,v)=2t+1$ for some $t>0$, then using Lemma 9, we have $d_S(u,v)\leqslant(9\cdot 5^{p-1}+6\cdot 5^{q-1}-1)(t-1)+(12\cdot 5^{p-1}+8\cdot 5^{q-1}-1)$. Hence, the stretch bound for even distances also holds. □

For concreteness, stretches given by Theorem 12 are compiled in the table of Fig. 9 for $k<10$. For $k=6$, which is not on the form 2^p+2^q-1, the next entry (7) must be taken.

4.2. Graphs with large minimum degree

It is known that sparser spanners exist whenever the minimum degree increases (cf. the concluding remark of [7]). In this paragraph, we show that graphs with minimum degree large enough enjoy an $O(1)$-spanner with only $O(n)$ edges, moreover computable with a fast deterministic distributed algorithm.

Let us first show that if a graph G has a ρ-dominating set of size γ (not necessarily independent), then in $O(\rho)$ time one can construct for G a $O(\rho)$-spanner with $n+O(\gamma^2)$ edges. Indeed, assuming we are given such a dominating set, the spanner can be constructed distributively in $O(\rho)$ time by first clustering the nodes of the graph around the nodes in the dominating set, and then by connecting every two neighboring clusters using one edge. The edges are composed of $n-\gamma$ edges for spanning all the regions, plus at most $\binom{\gamma}{2}$ edges for connecting every pair of adjacent regions. More formally we have:

Lemma 13. *For every integer $\rho\geqslant 1$, there exists a deterministic distributed algorithm that given a graph G with n nodes and a ρ-dominating set of size γ, constructs in $O(\rho)$ time for G a $(2\rho+1,2\rho)$-spanner of size at most $n-\gamma+\binom{\gamma}{2}$.*

Proof. The size of the spanner and the time complexity of the construction are clear from the above explanations. Let us detail the stretch analysis.

Consider any two nodes u,v at distance d in G, and consider a shortest path $P=p_0,\ldots,p_d$ connecting $u=p_0$ to $v=p_d$. Let $W=w_0,\ldots,w_t$ be a maximal subsequence of nodes in P such that $w_0=p_0$, $w_t=p_d$, and such that for every $0<i<d$, w_i belongs to a region that contains none of the w_j's with $j<i$. Finally, let c_i denote the center of the region containing w_i. Note that $d_S(c_i,w_i)\leqslant\rho$.

The key point is that there must exist an edge between the region of w_i and the region of w_{i+1}, for any $0\leqslant i<d$. Indeed, if not, then the subpath of P connecting w_i to w_{i+1} traverses a third region of some node p_m that appears between the nodes w_i and w_{i+1} in P: this contradicts the maximality of the sequence W.

In particular, there exists a path in S from c_i to c_{i+1} of length at most $2\rho+1$. Therefore, between $u=w_0$ and $v=w_t$ there exists in S a path of the form: $w_0\rightsquigarrow c_0\rightsquigarrow c_1\rightsquigarrow\cdots\rightsquigarrow c_t\rightsquigarrow w_t$, where $x\rightsquigarrow y$ denotes a shortest path in S going from x to y. The length of this path is at most $\rho+(2\rho+1)\cdot t+\rho\leqslant(2\rho+1)\cdot d+2\rho$ since $t\leqslant d$. Thus we have proved that $d_S(u,v)\leqslant(2\rho+1)\cdot d+2\rho$, i.e., S is a $(2\rho+1,2\rho)$-spanner. □

This lemma can be combined with the observation that if G has minimum degree $\delta\geqslant\sqrt{n\log n}$, then G has a 1-dominating set X of size $O(\sqrt{n\log n})$. Indeed, this can be proved using the following greedy algorithm [31]: one starts with $X=\varnothing$ and with the set of all radius-1 balls, $\mathcal{B}=\{N[v]\mid v\in V\}$, where $N[v]=\{u\in V\mid d_G(u,v)\leqslant 1\}$. Then, while $\mathcal{B}$ is non-empty, one selects a node $x\in V$ for X that belongs to the maximum number of balls in the current set $\mathcal{B}$. The set $\mathcal{B}$ is updated by removing all balls containing x. The constructed set X is a 1-dominating set and it can be shown that $|X|\leqslant n(1+\ln n)/\min_{v\in V}|N[v]|$ which is at most $O(\sqrt{n\log n})$ if $\delta\geqslant\sqrt{n\log n}$. Thus, the problem is to efficiently compute such 1-dominating set.

Unfortunately, no deterministic distributed implementation of the greedy algorithm faster than that $O(|X|)$ is known. A small ρ-dominating set can be computed much more efficiently in $O(\rho \log^* n)$ time by the algorithm of [26]. Unfortunately, its guaranteed size for X is only of $O(n/\rho)$. Finally, no algorithm is known to run in $o(\sqrt{n \log n})$ time for this problem.

However, using our technique, we obtain a spanner with only $O(n)$ edges, moreover with a better time complexity.

Theorem 14. *There exists a deterministic distributed algorithm that given a graph G with n nodes and minimum degree $\delta \geqslant \sqrt{n}$, constructs a $(5, 4)$-spanner for G with at most $3n/2$ edges in $O(\mathrm{MIS}(n))$ time.*

Proof. Notice that the size of a MIS of G^2 is at most $n/\delta \leqslant \sqrt{n}$. Moreover, a MIS of G^2 is a 2-dominating set. Since constructing a MIS for the logical graph G^2 takes $O(\mathrm{MIS}(n))$ time, we conclude by applying Lemma 13 for $\rho = 2$ and $\gamma \leqslant \sqrt{n}$. □

4.3. Randomized distributed implementation issues

In [32], Luby gives a simple and efficient randomized PRAM algorithm for computing a MIS in $O(\log n)$ expected time. Luby's algorithm can be turned to run in the distributed $\mathcal{LOCAL}$ model, and we obtain a distributed algorithm for computing an independent 1-dominating set which terminates within $O(\log n)$ expected time. We remark that upon termination of the algorithm, the constructed 1-dominating set is always correct, the randomization is only on the running time, i.e., it is a *Las Vegas* algorithm.

Thus, we obtain the following randomized version of Theorem 12:

Theorem 15. *There is a (*Las Vegas*) randomized distributed algorithm that given a graph G with n nodes and any fixed integer $k = 2^p + 2^q - 1$ with $p \geqslant q \geqslant 0$, constructs a spanner for G with $O(n^{1+1/k})$ edges in $O(\log n)$ expected time, with the same stretch properties than that in Theorem 12.*

Our (*Las Vegas*) randomized algorithms guarantee the stretch and the size bounds for the constructed spanners, while the $O(k)$ time (*Monte Carlo*) randomized algorithms [8] do not give any guarantee on the spanner size. This is of course achieved at the price of increasing the stretch factor of the spanner.

Recently, in [25], Kuhn et al. show that every packing problem can be approximated by a constant factor with high probability in $O(\log n)$ time in the $\mathcal{LOCAL}$ model. Therefore, the (*Monte Carlo*) algorithm of [25] implies a randomized constant approximation algorithm for the minimum 1-dominating set problem with $O(\log n)$ time. Thus, using Lemma 13, we obtain the following result (to be compared with Theorem 14 and [8]):

Theorem 16. *There exists a (*Monte Carlo*) randomized distributed algorithm that given a graph G with n nodes of minimum degree $\delta \geqslant \sqrt{n}$, constructs a $(3, 2)$-spanner for G in $O(\log n)$ time. The size is $O(n \log^2 n)$ edges with high probability. More generally, for a minimum degree δ graph, we obtain a $(3, 2)$-spanner with $O(n + (n \log n/\delta)^2)$ edges with high probability.*

Let us remark that the stretch given in Theorem 16 is best possible: there are n-node graphs with minimum degree $\delta = \Omega(\sqrt{n})$ for which any (α, β)-spanner with $\alpha + \beta < 5$ requires $\Omega(n^{3/2})$ edges. Indeed, it is known that there are n-node graphs with minimum degree at least $\frac{1}{2}\sqrt{n}$ and girth[2] 6. Thus, the deletion of any edge implies a stretch of at least 5 for its endpoints. Therefore, any (α, β)-spanner with size less than $\frac{1}{4}n\sqrt{n}$ requires $\alpha + \beta \geqslant 5$. Since, for[3] $\delta = \omega(n^{1/4} \log n)$ our spanner has $O(n + (n \log n/\delta)^2) = o(n\sqrt{n})$ edges, its stretch (α, β) must satisfied $\alpha + \beta \geqslant 5$.

5. Conclusion

In this paper we have considered deterministic distributed algorithms to construct low stretch and sparse spanners of unweighted arbitrary graphs. In particular, we have shown that $(3, 2)$-spanner with $O(n^{3/2})$ edges can be constructed in $n^{O(1/\sqrt{\log n})}$ time. Let us observe that $\log n < n^{1/\sqrt{\log n}}$ only for $n > 2^{4^2}$. In other words, deterministic distributed

[2] The length of its smallest cycle.

[3] The notation $f(n) = \omega(g(n))$ means that $g(n) = o(f(n))$.

$n^{1/\sqrt{\log n}}$ time algorithms might be competitive over randomized $\log n$ time algorithms for distributed systems of some thousand processors.

Recently, [13] have showed that $(3, 0)$-spanner with $O(n^{3/2})$ edges can be constructed in (deterministic) $O(\log n)$ time. This leads to the question of determining whether it is possible to extended for every $k > 2$ this latter result to a $(2k - 1, 0)$-spanner with $O(n^{1+1/k})$ edges in $O(f(k) \log n)$ time, for some function f. Finally, we leave open the natural question of determining the distributed time complexity of computing a $O(k)$-spanner with $O(n^{1+1/k})$ edges. Is the time complexity in $o(\log n)$? or even in $O(\log^* n)$?

Acknowledgement

The authors were supported by the project "PairAPair" of the ACI Masses de Données.

References

[1] B. Awerbuch, Complexity of network synchronization, Journal of the Association for Computing Machinery 32 (1985) 804–823.
[2] B. Awerbuch, Optimal distributed algorithms for minimum weight spanning tree, counting, leader election and related problems, in: 19th Annual ACM Symp. on Theory of Computing, STOC, ACM Press, 1987.
[3] B. Awerbuch, B. Berger, L.J. Cowen, D. Peleg, Near-linear cost sequential and distributed constructions of sparse neighborhood covers, in: 34th Annual IEEE Symposium on Foundations of Computer Science, FOCS, IEEE Computer Society Press, 1993.
[4] B. Awerbuch, B. Berger, L.J. Cowen, D. Peleg, Fast distributed network decompositions and covers, Journal of Parallel and Distributed Computing 39 (1996) 105–114.
[5] B. Awerbuch, B. Berger, L.J. Cowen, D. Peleg, Near-linear time construction of sparse neighborhood covers, SIAM Journal on Computing 28 (1) (1998) 263–277.
[6] B. Awerbuch, D. Peleg, Sparse partitions, in: 31th Annual IEEE Symposium on Foundations of Computer Science, FOCS, IEEE Computer Society Press, 1990.
[7] S. Baswana, T. Kavitha, K. Mehlhorn, S. Pettie, New constructions of (α, β)-spanners and purely additive spanners, in: 16th Symposium on Discrete Algorithms, SODA, ACM-SIAM, 2005.
[8] S. Baswana, S. Sen, A simple linear time algorithm for computing a $(2k - 1)$-spanner of $O(n^{1+1/k})$ size in weighted graphs, in: 30th International Colloquium on Automata, Languages and Programming, ICALP, in: Lecture Notes in Computer Science, vol. 2719, Springer, 2003.
[9] S. Baswana, S. Sen, Approximate distance oracles for unweighted graphs in $\tilde{O}(n^2)$ time, in: 15th Symposium on Discrete Algorithms, SODA, ACM-SIAM, 2004.
[10] E. Cohen, Fast algorithms for constructing t-spanners and paths with stretch t, SIAM Journal on Computing 28 (1) (1998) 210–236.
[11] R. Cole, U. Vishkin, Deterministic coin tossing with applications to optimal parallel list ranking, Information and Control 70 (1) (1986) 32–53.
[12] L.J. Cowen, Compact routing with minimum stretch, Journal of Algorithms 38 (1) (2001) 170–183.
[13] B. Derbel, C. Gavoille, D. Peleg, Deterministic distributed construction of linear stretch spanners in polylogarithmic time, in: 21st International Symposium on Distributed Computing, DISC, in: Lecture Notes in Computer Science, vol. 4731, Springer, 2007.
[14] B. Derbel, M. Mosbah, A. Zemmari, Fast distributed graph partition and application, in: 20th IEEE International Parallel & Distributed Processing Symposium, IPDPS, IEEE Computer Society Press, 2006.
[15] T. Eilam, C. Gavoille, D. Peleg, Compact routing schemes with low stretch factor, Journal of Algorithms 46 (2003) 97–114.
[16] M. Elkin, Computing almost shortest paths, in: 20th Annual ACM Symp. on Principles of Distributed Computing, PODC, ACM Press, 2001.
[17] M. Elkin, A faster distributed protocol for constructing a minimum spanning tree, in: 15th Symp. on Discrete Algorithms, SODA, ACM-SIAM, 2004.
[18] M. Elkin, Unconditional lower bounds on the time-approximation tradeoffs for the distributed minimum spanning tree problems, in: 36th Annual ACM Symp. on Theory of Computing, STOC, ACM Press, 2004.
[19] M. Elkin, D. Peleg, $(1 + \epsilon, \beta)$-spanner constructions for general graphs, SIAM Journal on Computing 33 (3) (2004) 608–631.
[20] M. Elkin, J. Zhang, Efficient algorithms for constructing $(1 + \epsilon, \beta)$-spanners in the distributed and streaming models, in: 23rd Annual ACM Symposium on Principles of Distributed Computing, PODC, ACM Press, 2004.
[21] C. Gavoille, D. Peleg, S. Pérennès, R. Raz, Distance labeling in graphs, Journal of Algorithms 53 (1) (2004) 85–112.
[22] A.V. Goldberg, S.A. Plotkin, G.E. Shannon, Parallel symmetry-breaking in sparse graphs, SIAM Journal on Discrete Mathematics 1 (4) (1988) 434–446.
[23] F. Kuhn, T. Moscibroda, T. Nieberg, R. Wattenhofer, Fast deterministic distributed maximal independent set computation on growth-bounded graphs, in: 19th International Symposium on Distributed Computing, DISC, in: Lecture Notes in Computer Science, vol. 3724, Springer, 2005.
[24] F. Kuhn, T. Moscibroda, R. Wattenhofer, What cannot be computed locally! in: 23rd Annual ACM Symposium on Principles of Distributed Computing, PODC, ACM Press, 2004.
[25] F. Kuhn, T. Moscibroda, R. Wattenhofer, The price of being near-sighted, in: 17th Symposium on Discrete Algorithms, SODA, ACM-SIAM, 2006.
[26] S. Kutten, D. Peleg, Fast distributed construction of small k-dominating sets and applications, Journal of Algorithms 28 (1) (1998) 40–66.

[27] N. Linial, Distributive graph algorithms — Global solutions from local data, in: 28th Annual IEEE Symposium on Foundations of Computer Science, FOCS, IEEE Computer Society Press, 1987.
[28] N. Linial, Locality in distributed graphs algorithms, SIAM Journal on Computing 21 (1) (1992) 193–201.
[29] Z. Lotker, B. Patt-Shamir, E. Pavlov, D. Peleg, Minimum-weight spanning tree construction in $O(\log \log n)$ communication rounds, SIAM Journal on Discrete Mathematics 35 (1) (2005) 120–131.
[30] Z. Lotker, B. Patt-Shamir, D. Peleg, Distributed MST for constant diameter graphs, in: 20th Annual ACM Symposium on Principles of Distributed Computing, PODC, ACM Press, 2001.
[31] L. Lovász, On the ratio of optimal integral and fractional covers, Discrete Mathematics 13 (1975) 383–390.
[32] M. Luby, A simple parallel algorithm for the maximal independent set problem, SIAM Journal on Computing 15 (4) (1986) 1036–1053.
[33] S. Moran, S. Snir, Simple and efficient network decomposition and synchronization, Theoretical Computer Science 243 (1–2) (2000) 217–241.
[34] A. Panconesi, A. Srinivasan, On the complexity of distributed network decomposition, Journal of Algorithms 20 (2) (1996) 356–374.
[35] D. Peleg, Distributed Computing: A Locality-Sensitive Approach, in: SIAM Monographs on Discrete Mathematics and Applications, 2000.
[36] D. Peleg, V. Rubinovich, A near-tight lower bound on the time complexity of distributed minimum-weight spanning tree construction, SIAM Journal on Computing 30 (5) (2000) 1427–1442.
[37] D. Peleg, J.D. Ullman, An optimal synchornizer for the hypercube, SIAM Journal on Computing 18 (4) (1989) 740–747.
[38] D. Peleg, E. Upfal, A trade-off between space and efficiency for routing tables, Journal of the ACM 36 (3) (1989) 510–530.
[39] L.D. Penso, C.B. Valmir, A distributed algorithm to find k-dominating sets, Discrete Applied Mathematics 141 (1–3) (2004) 243–253.
[40] L. Roditty, M. Thorup, U. Zwick, Roundtrip spanners and roundtrip routing in directed graphs, in: 13th Symposium on Discrete Algorithms, SODA, ACM-SIAM, 2002.
[41] L. Roditty, M. Thorup, U. Zwick, Deterministic constructions of approximate distance oracles and spanners, in: 32nd International Colloquium on Automata, Languages and Programming, ICALP, in: Lecture Notes in Computer Science, vol. 3580, 2005.
[42] M. Thorup, U. Zwick, Compact routing schemes, in: 13th Annual ACM Symposium on Parallel Algorithms and Architectures, SPAA, ACM Press, 2001.
[43] M. Thorup, U. Zwick, Approximate distance oracles, Journal of the ACM 52 (1) (2005) 1–24.
[44] R. Wenger, Extremal graphs with no C^4's, C^6's, or C^{10}'s, Journal of Combinatorial Theory, Series B 52 (1) (1991) 113–116.