Local Computation of Nearly Additive Spanners

Bilel Derbel[1,⋆], Cyril Gavoille[2,⋆⋆], David Peleg[3,⋆⋆⋆], and Laurent Viennot[4,†]

[1] Laboratoire d'Informatique Fondamentale de Lille (LIFL),
Université des Sciences et Technologies de Lille, France
bilel.derbel@lifl.fr
[2] Laboratoire Bordelais de Recherche en Informatique (LaBRI),
Université de Bordeaux, France
gavoille@labri.fr
[3] Department of Computer Science and Applied Mathematics,
The Weizmann Institute of Science, Rehovot, Israel
david.peleg@weizmann.ac.il
[4] INRIA, University Paris 7, France
Laurent.Viennot@inria.fr

Abstract. An (α, β)-spanner of a graph G is a subgraph H that approximates distances in G within a multiplicative factor α and an additive error β, ensuring that for any two nodes u, v, $d_H(u, v) \leq \alpha \cdot d_G(u, v) + \beta$. This paper concerns algorithms for the distributed deterministic construction of a sparse (α, β)-spanner H for a given graph G and distortion parameters α and β. It first presents a generic distributed algorithm that in constant number of rounds constructs, for every n-node graph and integer $k \geq 1$, an (α, β)-spanner of $O(\beta n^{1+1/k})$ edges, where α and β are constants depending on k. For suitable parameters, this algorithm provides a $(2k-1, 0)$-spanner of at most $kn^{1+1/k}$ edges in k rounds, matching the performances of the best known distributed algorithm by Derbel et al. (PODC '08). For $k = 2$ and constant $\varepsilon > 0$, it can also produce a $(1+\varepsilon, 2-\varepsilon)$-spanner of $O(n^{3/2})$ edges in constant time. More interestingly, for every integer $k > 1$, it can construct in constant time a $(1+\varepsilon, O(1/\varepsilon)^{k-2})$-spanner of $O(\varepsilon^{-k+1} n^{1+1/k})$ edges. Such deterministic construction was not previously known. The paper also presents a second generic deterministic and distributed algorithm based on the construction of small dominating sets and maximal independent sets. After computing such sets in sub-polynomial time, it constructs at its best a $(1+\varepsilon, \beta)$-spanner with $O(\beta n^{1+1/k})$ edges, where $\beta = k^{\log(\log k/\varepsilon)+O(1)}$. For $k = 3$, it provides a $(1+\varepsilon, 6-\varepsilon)$-spanner with $O(\varepsilon^{-1} n^{4/3})$ edges. The additive terms $\beta = \beta(k, \varepsilon)$ in the stretch of our constructions yield the best trade-off currently known between k and ε, due to Elkin and Peleg (STOC '01). Our distributed algorithms are rather short, and can be viewed as a unification and simplification of previous constructions.

⋆ Supported by the équipe-projet INRIA "DOLPHIN".
⋆⋆ Supported by the ANR-project "ALADDIN", and the équipe-projet INRIA "CÉPAGE".
⋆⋆⋆ Supported by grants from the Israel Science Foundation and the Minerva Foundation.
† Supported by the ANR-project "ALADDIN", and the équipe-projet INRIA "GANG".

I. Keidar (Ed.): DISC 2009, LNCS 5805, pp. 176–190, 2009.

1 Introduction

Applications for networks. Sparse spanners are motivated by routing protocols used in practical networks, where fast construction of a "skeleton" of the underlying network topology is crucial. As recently shown in [1], spanners and their variants can be efficiently used for routing in ad-hoc networks in view of the IETF standardized OLSR routing protocol [2].

Sparse spanners, as introduced by Peleg et al. [3,4], and implicitly used in [5], are key ingredients of various distributed applications, e.g., synchronizers [6], computing almost shortest paths in distributed networks [7,8], or distance oracles [9,10,11,12]. Spanners have also found applications in approximation algorithms for geometric spaces [13], and for solving linear systems [14]. In all of those problems, the quality of the spanners used directly impacts the quality of the solutions.

Spanners and their variants. Given an undirected unweighted graph G, let $d_G(u,v)$ denote the distance between u and v in G. An (α,β)-*spanner* of G is a spanning subgraph H of G such that $d_H(u,v) \leq \alpha \cdot d_G(u,v) + \beta$ for every two nodes u, v. There are several variations on the concept of spanners. A spanning H that is not restricted to be a subgraph of G is called an (α,β)-*emulator* of G [15,16]. A subgraph H of G that must preserve distances larger than d only is called a d-*preserver* for G [17]. Other recent developments can be found in [18]. The paper will not discuss any of these variants, as well as extensions for digraphs [19].

Constructing sparse spanners. There is an abundant literature on spanners and related combinatorial objects, which is surveyed, e.g., by Pettie in [20]. It is well-known that every n-node graph has a $(2k-1,0)$-spanner with $O(n^{1+1/k})$ edges, which can be obtained by modification of the Kruskal's minimum spanning tree algorithm [21]. Moreover, according to Erdös-Simonovits Girth Conjecture [22,23], it is believed that every (α,β)-spanner with $\alpha+\beta < 2k+1$ must have $\Omega(n^{1+1/k})$ edges for some worst-case graphs. The lower bound suggests that (α,β)-spanners such that $\alpha+\beta = 2k-1$ and $\alpha < 2k-1$ with $O(n^{1+1/k})$ edges may exist for all graphs. Indeed, for $\alpha = 1$, $(1,2)$-spanner of size $O(n^{3/2})$ [24], and $(1,6)$-spanner of size $O(n^{4/3})$ [25] exist for all graphs. It is not known whether $(1,4)$-spanners with $O(n^{4/3})$ edges exist, or if $(1,O(1))$-spanner with $o(n^{4/3})$ edges can exist. Woodruff [26] proved, for every $k>0$, that every $(1,2k-2)$-spanner requires $\Omega(n^{1+1/k})$ edges in the worst-case, independently of the Erdös-Simonovits Girth Conjecture. For $\alpha = 1+\varepsilon$, and for small $\varepsilon > 0$, Elkin and Peleg [27] showed that $(1+\varepsilon,\beta)$-spanners with $O(\beta n^{1+1/k})$ edges exist, where[1] $\beta = k^{\log(\log k/\varepsilon)+O(1)}$. Thorup and Zwick [16] showed that $(1+\varepsilon, O(1/\varepsilon)^{k-2})$-spanners with $O(kn^{1+1/k})$ edges exist. The stretch is worse than in the Elkin-Peleg construction, however it holds simultaneously for all ε. Note that their construction is not local as it possibly involves collaboration between nodes at distance $\Omega(n)$. (This occurs, for instance, for the n-node path.) Pettie [20,28] addressed the problem of constructing spanners of *linear* size. E.g., $(1,\tilde{O}(n^{9/16}))$-spanners[2] and $(O(1),\tilde{O}(1))$-spanners with $O(n)$ edges are presented in [20].

[1] All logarithms are in base two.

[2] The notation $\tilde{O}(f(n))$ stands for $f(n)\log^{O(1)} n$.

Distributed algorithms. Efficiently constructing sparse spanners by distributed algorithms is clearly important for network applications. Indeed, distributed algorithms for constructing $(2k-1,0)$-spanners with $O(kn^{1+1/k})$ edges exist. By the above discussion, these constructions are essentially optimal in size and stretch (distortion). A randomized algorithm achieving this performance (with guarantees on the stretch and expected size) has been presented in [29]. It has been recently shown in [30] that randomization is actually not required, and that $(2k-1,0)$-spanners with $O(kn^{1+1/k})$ edges can be constructed in k rounds. Interestingly, allowing $2k$ additional rounds, the algorithm can work without any knowledge of n, and still provide the same guarantee on the maximum spanner size.

When k tends to $\log n$, such constructions achieve $\Omega(n \log n)$ size only. A series of (randomized) constructions producing linear or near-linear size has been presented in [28]. At its sparsest level, $(1+\varepsilon,\beta)$-spanners with $n \cdot (\varepsilon^{-1} \log\log n)^{O(1)}$ edges are constructed in $O(\beta)$ time, where β is in the form $(\varepsilon^{-1} \log\log n)^{O(\log\log n)}$.

For $(1+\varepsilon,\beta)$-spanners, only few distributed constructions are known. Actually, it has been proved in [30,28] that $(1, f(k))$-spanners (for some function $f(k)$ of k), which are known to exist for $k=2$ and $k=3$, cannot be constructed quickly (say, in polylog time). So the best polylogarithmic-time distributed constructions one may hope for will yield $(1+\varepsilon, f(k))$-spanners. For $k=2$, a $(1+\varepsilon,2)$-spanner with $O(\varepsilon^{-1}n^{3/2})$ edges is constructed in [30] in $O(\varepsilon^{-1})$ time.

There were previous attempts to devise a distributed implementation of the Elkin-Peleg construction of $(1+\varepsilon,\beta)$-spanners with $\beta = \beta(k,\varepsilon)$ and arbitrary k,ε. However, as pointed out by several authors [31,20], the resulting constructions, while achieving the goal of demonstrating the existence of sparse $(1+\varepsilon,\beta)$-spanners, can hardly serve as a basis for an efficient algorithm. (We refer the reader to [31] for a discussion on the technical reasons for the difficulty of implementing these constructions in a distributed setting.) Nevertheless, Elkin and Zhang [32] proposed a distributed implementation of $(1+\varepsilon,\beta)$-spanners (albeit through a very complicated algorithm). The trade-off for β is worse than the one of [33] by a factor of roughly $k^{\log k}$ (more precisely, in [32], $\beta = O((k\log k)/\varepsilon)^{\log k} = k^{\log k} \cdot k^{\log(\log k/\varepsilon)+O(1)}$), and the algorithm is randomized.

Our results. In this article, we come up with an alternative construction of sparse $(1+\varepsilon,\beta)$-spanners, and demonstrate that the new construction leads to significant improvements in the current state-of-the-art for the problem of computing almost shortest paths in distributed settings. It positively answers Pettie's open question [28] concerning the deterministic construction of additive spanners.

We present two algorithms. The first (in Section 2) constructs sparse spanners in *constant* time, for fixed k and ε. In the spirit of Pettie's constructions [20], our algorithm is generic. Depending on the parameters, it can achieve, for instance, a $(2k-1,0)$-spanner with $O(kn^{1+1/k})$ edges, or a $(1+\varepsilon, O(1/\varepsilon)^{k-2})$-spanner with $O(\varepsilon^{-k+1}n^{1+1/k})$ edges. More specifically, it can produce a $(1+\varepsilon, 2-\varepsilon)$-spanner with $O(\varepsilon^{-1}n^{3/2})$ edges. Note that this latter construction is optimal even in the sequential sense, since the absolute lower bound discussed above implies that $\alpha+\beta = (1+\varepsilon)+(2-\varepsilon) \geq 3$ for such a number of edges. Other trade-offs produced by our algorithm are summarized in the table of Section 2.2. Finally,

it has the extra feature that it does not require the nodes to know the value of n, and still guarantees the desired size. This first contribution provides a positive answer to Pettie's open question [28].

Our second construction (Section 3) runs in sub-polynomial time, and relies on the deterministic computation of maximal independent sets, which is known to be difficult in the distributed setting [34]. Similarly to the first algorithm, it is generic and can be parameterized to produce a new family of spanners. In particular (see the table in Section 3.3 for more details), it provides a $(1+\varepsilon, \beta)$-spanner with $O(\beta n^{1+1/k})$ edges in sub-polynomial time, where $\beta = k^{\log(\log k/\varepsilon)+O(1)}$. This matches the performance of the best existential (sequential) constructions of [33]. As a particular case, our algorithm can also produce a $(1+\varepsilon, 8-\varepsilon)$-spanner with $O(\varepsilon^{-1}n^{4/3})$ edges, and a specific construction, with the same number of edges, actually provides a $(1+\varepsilon, 6-\varepsilon)$-spanner (Subsection 3.4). We also observe that using a Las Vegas algorithm for selecting a maximal independent set, our algorithms can run in poly-logarithmic time while achieving the best known stretches. Finally, our implementation is considerably simpler than that of [32].

In this paper we consider the classical $\mathcal{LOCAL}$ model of computation [35,36], where in each time unit a node can send any amount of information to its neighbors and perform any amount of local computations. Although the issue of message size may be important (see e.g., [32,28]), we do not address it in this paper, and leave open the question of deterministically constructing similar spanners with low message complexity.

Open questions. We leave open two main questions for further study. First, can the performances of the second construction be achieved deterministically in polylog time without the bottleneck of "breaking symmetry"? and with short messages? Second, do $(1+\varepsilon, f(k))$-spanners with at most $g(k)\cdot n^{1+1/k}$ edges exist for all graphs, for fixed $\varepsilon > 0$ and for some $f(k) = k^{O(1)}$? or even $f(k) = O(k)$? As far as we know, the best upper bound is $f(k) \leq k^{\log\log k+O(1)}$.

2 A Local Algorithm

2.1 Description of Algorithm Local-Span

A *distance sequence* is a sequence of strictly positive integers. Given a distance sequence $\rho_1, \ldots, \rho_k$, denote its partial sums by

$$\rho[i,j] \;=\; \begin{cases} \rho_i + \cdots + \rho_j, & \text{if } i \leq j, \\ 0, & \text{if } i > j. \end{cases} \tag{1}$$

For every subgraph H of G, denote by $B_H(u,\rho)$ the ball of radius ρ in H centered at u. The subscript is omitted when $H = G$ is clear from the context.

The deterministic distributed algorithm Local-Span is presented next. Informally, the algorithm operates in k iterations, during which each node u builds a cluster $R(u)$ around itself. At any stage, the subgraph H consists of all the edges selected to the spanner so far. This H enjoys the property that at any stage and

for any node u, the subgraph of H induced by the nodes of the cluster $R(u)$ is connected. In iteration i, the "target radius" of the constructed cluster is $\rho[1,i]$. Every node u learns the clusters $R(v)$ of all the nodes v in its ρ_i-neighborhood, $B(u,\rho_i)$. Of those nodes, it keeps in the set $W(u)$ all the candidates to join its cluster $R(u)$. In an internal loop, it selects up to σ such candidates w from $W(u)$ and adds their clusters $R(w)$ to its own cluster $R(u)$, by adding to H a shortest path connecting w and u.

Input: a graph $G=(V,E)$, a distance sequence $\rho_1,\ldots,\rho_k$
Output: a spanner $H=\bigcup_{u\in V} H(u)$ of G

Set σ to any value in the range $[\max_{v\in B(u,\rho[1,k])} |B(v,\rho[1,k])|^{1/k}, n^{1/k}]$
$R(u) := \{u\}$ /* cluster around u */
$F(u) :=$ False /* termination flag */
$H(u) := (\{u\},\varnothing)$ /* spanner edges selected by u */
for $i := 1$ *to* k **do**
  Node u sends $R(u), F(u)$ to all nodes in $B(u,\rho_i)$,
  and receives $R(v), F(v)$ from all $v \in B(u,\rho_i)$
  $W(u) := B(u,\rho_i) \setminus \{v \mid F(v) =$ True$\}$ /* candidate nodes to be covered */
  $\ell := 0$
  while $\exists w \in W(u)$ *and* $\ell < \sigma$ **do**
    (a) Pick $w \in W(u)$ such that $d_G(u,w)$ is minimal
    (b) Add a shortest path in G from u to w to $H(u)$
    (c) Add $R(w)$ to $R(u)$
    (d) $W(u) := W(u) \setminus \{v \in W(u) \mid R(v) \cap R(w) \neq \varnothing\}$
    (e) $\ell := \ell + 1$
  if $W(u) = \varnothing$ **then** $F(u) :=$ True **else** $F(u) :=$ False

Algorithm 1. Algorithm Local-Span - Code for a node u

2.2 Results

Theorem 1. *Algorithm* Local-Span *computes, for every n-node graph G and distance sequence $\rho_1,\ldots,\rho_k$, a spanner H of at most $\rho[1,k]\cdot n^{1+1/k}$ edges. The stretch of H and the time complexity of* Local-Span *are summarized in the following table.*

stretch	size	time	parameters
$(2k-1,0)$	$k\cdot n^{1+1/k}$	$O(k)$	$\rho_1=\cdots=\rho_k=1$
$(1+\varepsilon,2-\varepsilon)$	$(1+\frac{2}{\varepsilon})\cdot n^{3/2}$	$O(\varepsilon^{-1})$	$\rho_1=1,\rho_2=2/\varepsilon$ $\varepsilon\in(0,2]$
$(1+\varepsilon,4(1+\frac{4}{\varepsilon})^{k-2}-\varepsilon)$	$(1+\frac{4}{\varepsilon})^{k-1}\cdot n^{1+1/k}$	$O((1+\frac{4}{\varepsilon})^{k-1})$	$\rho_1=1,\rho_i=\frac{4}{\varepsilon}(1+\frac{4}{\varepsilon})^{i-2}$ $\varepsilon\in(0,4]$
$(5,2^k-4)$	$5^{k-1}\cdot n^{1+1/k}$	$O(5^k)$	$\hookrightarrow$ with $\varepsilon=4$
$(3,4\cdot 3^{k-2}-2)$	$3^{k-1}\cdot n^{1+1/k}$	$O(3^k)$	$\hookrightarrow$ with $\varepsilon=2$

The correctness of algorithm LOCAL-SPAN and Theorem 1 are proved in the next section.

2.3 Analysis of Algorithm LOCAL-SPAN

Let $H_i(u)$, $R_i(u)$, $F_i(u)$, and $W_i(u)$ denote the values of $H(u)$, $R(u)$, $F(u)$, and $W(u)$, respectively, at the end of iteration i. The parameter u is omitted from these notations when u is clear from the context.

Proposition 1. *Algorithm* LOCAL-SPAN *has time complexity* $\rho[1,k]$ *if* n *is known to each node, and* $3\rho[1,k]$ *otherwise.*

Proof. At Step i, a node communicates with other nodes at distance at most ρ_i. So after $\rho[1,k]$ rounds the algorithm ends. If n is known, then σ can be set immediately to $n^{1/k}$. If it is not, then σ can be set to $\max_{v\in B(u,\rho[1,k])} |B(v,\rho[1,k])|^{1/k}$, and calculating this value requires $2\rho[1,k]$ extra rounds. □

Proposition 2. *The resulting spanner* H *has at most* $\rho[1,k]\cdot n^{1+1/k}$ *edges.*

Proof. Each node u adds to H, in each iteration i of its main loop, up to $\sigma=\sigma(u)$ paths of length at most ρ_i. Hence the overall contribution of u to the spanner consists of at most $\sigma\cdot\rho[1,k]$ edges. The bound follows as $\sigma\le n^{1/k}$. □

The following proposition is proved by induction on i.

Proposition 3. *For all* $i\ge 0$, $R_i(u)\subseteq B_{H_i}(u,\rho[1,i])$.

Proposition 4. *Let* x *be a node at distance at most* ρ_i *from* u *and satisfying* $F_{i-1}(x)=$ FALSE. *If* $F_i(u)=$ TRUE *then* $d_{H_i}(u,x)\le d_G(u,x)+2\rho[1,i-1]$.

Proof. As $F_{i-1}(x)=$ FALSE, x is in $W(u)$ before the while loop of iteration i. As $F_i(u)=$ TRUE, i.e., $W_i(u)=\varnothing$, there must exist some $w\in W(u)$ such that $R_{i-1}(w)\cap R_{i-1}(x)\neq\varnothing$. Consider the first vertex w satisfying this (eventually $w=x$) and let $z\in R_{i-1}(w)\cap R_{i-1}(x)$. By the triangle inequality,

$$\begin{aligned} d_{H_i}(u,x) &\le d_{H_i}(u,w)+d_{H_i}(w,z)+d_{H_i}(z,x)\\ &\le d_{H_i}(u,w)+d_{H_{i-1}}(w,z)+d_{H_{i-1}}(z,x) \end{aligned} \tag{2}$$

The choice of w and step (b) in the while loop imply that $d_{H_i}(u,w)=d_G(u,w)\le d_G(u,x)$. In addition, Proposition 3, applied to $R_{i-1}(w)$ and $R_{i-1}(x)$, implies that $d_{H_{i-1}}(w,z)\le\rho[1,i-1]$ and $d_{H_{i-1}}(z,x)\le\rho[1,i-1]$. Hence (2) yields $d_{H_i}(u,x)\le d_G(u,x)+2\rho[1,i-1]$.

In the special case $i=1$, we indeed have $d_{H_1}(u,x)=d_G(u,x)$ since $W_1(u)=\varnothing$ implies that $R_1(u)$ contains all nodes in $B(u,\rho_1)$. Indeed, as no nodes v satisfy $R_0(v)=\varnothing$, $W(u)=B(u,\rho_1)$ before the while loop. We then obtain $d_{H_1}(u,x)\le d_G(u,x)+2\rho[1,0]$ as claimed because $\rho[1,0]=0$. □

Define $\bar{R}_i(u)$ as the union of the sets $R(w)$ added to $R(u)$ during the while loop of iteration i.

Proposition 5. *If* $F_i(u) = \text{FALSE}$*, then*

$$|R_i(u)| \;\geq\; |\bar{R}_i(u)| \;\geq\; \max_{v \in B(u,\rho[i+1,k])} |B(v,\rho[1,k])|^{i/k}.$$

Proof. By induction on i. The assertion is satisfied for $i = 0$ as $|R_0(u)| = 1$. Consider the sets $R_{i-1}(w)$ added to $R(u)$ in the while loop of iteration i. The inductive hypothesis implies

$$\begin{aligned} |R_{i-1}(w)| &\geq \max_{v \in B(w,\rho[i,k])} |B(v,\rho[1,k])|^{(i-1)/k} \\ &\geq \max_{v \in B(u,\rho[i+1,k])} |B(v,\rho[1,k])|^{(i-1)/k} \end{aligned}$$

since $d_G(u,w) \leq \rho_i$. If $F_i(u) = \text{FALSE}$, then

$$\sigma \;=\; \max_{v \in B(u,\rho[1,k])} |B(v,\rho[1,k])|^{1/k} \;\geq\; \max_{v \in B(u,\rho[i+1,k])} |B(v,\rho[1,k])|^{1/k}$$

sets are added to $R(u)$. As these sets are disjoint, the size of $\bar{R}_i(u)$ is thus at least $\max_{v \in B(u,\rho[i+1,k])} |B(v,\rho[1,k])|^{i/k}$. □

Proposition 6. *For all* u*,* $F_k(u) = \text{TRUE}$*.*

Proof. Suppose, towards contradiction, that $F_k(u) = \text{FALSE}$ for some u. Proposition 5 then implies

$$|\bar{R}_k(u)| \;\geq\; \max_{v \in B(u,\rho[k+1,k])} |B(v,\rho[1,k])|^{k/k} \;=\; |B(u,\rho[1,k])|.$$

Moreover, Proposition 3 implies $\bar{R}_k(u) \subseteq B(u,\rho[1,k])$. We thus deduce $\bar{R}_k(u) = B(u,\rho[1,k])$. In particular, $\bar{R}_k(u)$ contains $B(u,\rho_k)$. As every node that is added to $R(u)$ is immediately removed from $W(u)$, all $B(u,\rho_k)$ is removed from $W(u)$ and necessarily $W_k(u) = \varnothing$. This is in contradiction with the fact that $F_k(u) = \text{FALSE}$. □

Proposition 7. *For every* u, v*, we have*

$$d_H(u,v) \;\leq\; (1+\varepsilon) \cdot d_G(u,v) + 4\rho[1,k-1] - \varepsilon$$

where $\varepsilon = \max_{1 \leq i \leq k} \{4\rho[1,i-1]/\rho_i\}$*.*

Proof. We prove this by induction on $d_G(u,v)$. The claim is obviously satisfied for $d_G(u,v) = 0$. Now consider u and v at distance $\delta = d_G(u,v)$ and suppose that the property is verified for any pair of nodes u', v' such that $d_G(u',v') < \delta$. Consider a shortest path P from u to v in G. Let x_i denote the vertex at distance ρ_i from u on P (we set $x_0 = u$ for $i = 0$ and $x_i = v$ if $\rho_i \geq d_G(u,v)$). Let P_i denote the sub-path of P from u to x_i.

Consider the lowest value of i such that $F_i(y) = \textsc{True}$ for all $y \in P_i$. (Note that $i \leq k$ by Proposition 6). Then $F_{i-1}(y) = \textsc{False}$ for some $y \in P_{i-1}$. As $F_i(u) = \textsc{True}$ and $F_i(x_i) = \textsc{True}$, Proposition 4 implies $d_{H_i}(u,y) \leq d_G(u,y) + 2\rho[1,i-1]$ and $d_{H_i}(x_i,y) \leq d_G(x_i,y) + 2\rho[1,i-1]$. By the triangle inequality, $d_{H_i}(u,x_i) \leq d_{H_i}(u,y) + d_{H_i}(y,x_i) \leq d_G(u,y) + d_G(y,x_i) + 4\rho[1,i-1] \leq d_G(u,x_i) + 4\rho[1,i-1]$.

In case $x_i = v$, we thus obtain $d_H(u,v) \leq (1+\varepsilon) \cdot d_G(u,v) + 4\rho[1,k-1] - \varepsilon d_G(u,v) \leq (1+\varepsilon) \cdot d_G(u,v) + 4\rho[1,k-1] - \varepsilon$, since $d_G(u,v) \geq 1$.

In case $x_i \neq v$ (i.e., $d_G(u,v) > \rho_i$), we have $d_G(u,x_i) = \rho_i$ and $d_H(u,x_i) \leq \rho_i + 4\rho[1,i-1]$. By the choice of ε, $\varepsilon \geq 4\rho[1,i-1]/\rho_i$, and thus $d_H(u,x_i) \leq \rho_i + \varepsilon\rho_i = (1+\varepsilon)\rho_i$. By the induction hypothesis, $d_H(x_i,v) \leq (1+\varepsilon)\cdot(\delta - \rho_i) + 4\rho[1,k-1] - \varepsilon$; the desired inequality for u,v follows. □

Consider the sequence defined by $\rho_1 = 1$ and $\rho_i = \alpha\,(1+\alpha)^{i-2}$ for $i \geq 2$, which is a distance sequence for every $\alpha \geq 1$. We have

$$\rho[1,i-1] \;=\; 1 + \alpha \sum_{j=0}^{i-3} (1+\alpha)^j \;=\; 1 + \alpha \frac{(1+\alpha)^{i-2} - 1}{(1+\alpha) - 1} \;=\; (1+\alpha)^{i-2} = \rho_i/\alpha.$$

This yields $\varepsilon = \max_i \{4\rho[1,i-1]/\rho_i\} = 4/\alpha$. By Proposition 7, for every $\varepsilon \in (0,4]$, $d_H(u,v) \leq (1+\varepsilon) \cdot d_G(u,v) + 4\,(1+4/\varepsilon)^{k-2} - \varepsilon$. By Proposition 2, the number of edges of H is no more than $(1+4/\varepsilon)^{k-1} \cdot n^{1+1/k}$. For instance, for $\varepsilon = 2, 3$ or 4, we get for H the stretches $(2, 4 \cdot 5^{k-2} - 1)$, $(3, 4 \cdot 3^{k-2} - 2)$, and $(5, 2^k - 4)$, respectively.

We can obtain a better analysis when $\rho_1 = \cdots = \rho_{k-1}$.

Proposition 8. *If $\rho_1 = \cdots = \rho_{k-1}$, then for all u,v, we have $d_H(u,v) \leq (1+\varepsilon) \cdot d_G(u,v) + 2\rho[1,k-1] - \varepsilon$ where $\varepsilon = \max_{1 \leq i \leq k} \{2\rho[1,i-1]/\rho_i\}$.*

Proof. In the proof of Proposition 7, consider the highest value of i such that $F_i(u) = \textsc{False}$ and $F_i(x_1) = \textsc{False}$. Proposition 6 implies $i < k$. In case $i = k-1$, Proposition 4 implies $d_H(u,x_k) \leq d_G(u,x_k) + 2\rho[1,k-1]$ since $F_k(x_k) = \textsc{True}$. In case $i < k-1$, we have $F_{i+1}(u) = \textsc{True}$ or $F_{i+1}(x_1) = \textsc{True}$. Proposition 4 then implies $d_H(u,x_1) \leq d_G(u,x_1) + 2\rho[1,i]$. In both cases, we have $d_H(u,x_{i+1}) \leq d_G(u,x_{i+1}) + 2\rho[1,i]$. We can then conclude similarly to the proof of Proposition 7. □

Consider the distance sequence $\rho_1 = \cdots = \rho_k = 1$. Then $\rho[1,k-1] = k-1$ and $\varepsilon = \max_{1 \leq i \leq k} \{2\rho[1,i-1]/\rho_i\} = 2(k-1)$. By Proposition 7, it follows that $d_H(u,v) \leq (2k-1) \cdot d_G(u,v)$. The number of edges is $k \cdot n^{1+1/k}$.

For $k = 2$, consider the distance sequence $\rho_1 = 1$ and $\rho_2 = 2/\varepsilon$, for every $\varepsilon \in (0,2]$. We have $\max_{1 \leq i \leq k} \{2\rho[1,i-1]/\rho_i\} = \max\{2\rho[1,0]/1, 2\rho[1,1]/(2/\varepsilon)\} = \varepsilon$, and also $\rho[1,k-1] = 1$, which yields, for every $\varepsilon \in (0,2]$, a stretch of $(1+\varepsilon, 2-\varepsilon)$ for $(1+2/\varepsilon) \cdot n^{3/2}$ edges.

3 An Algorithm Based on Independent Dominating Sets

3.1 Definitions

Let us consider a graph $G = (V, E)$. A triple (v, S, T) is called a *cluster* if $S \subseteq V$, $v \in S$, and T is a tree of G spanning[3] S. The node v is called the *center* of the cluster. Two clusters (v, S_v, T_v) and (w, S_w, T_w) are *disjoint* if $S_v \cap S_w = \varnothing$.

Let $\mathcal{C}$ be a collection of clusters of G. If the clusters of $\mathcal{C}$ are pairwise disjoint, we say that $\mathcal{C}$ is a *partition* of G. We denote by center($\mathcal{C}$) the set of centers of all clusters of $\mathcal{C}$. For $v \in$ center($\mathcal{C}$), denote by S_v and T_v the subset and tree such that (v, S_v, T_v) is a cluster of $\mathcal{C}$.

For two node sets S, S', let $d_G(S, S') = \min\{d_G(v, v') \mid v \in S, v' \in S'\}$. Denote by $G_\rho(\mathcal{C})$ the graph whose vertex set is center($\mathcal{C}$), and whose edge set is the set of all pairs of centers u, v such that $d_G(S_v, S_w) \leq \rho$.

Denote by G^2 the graph obtained from G by adding an edge between every two nodes at distance 2 in G. Given a set W of nodes, let $G[W]$ denote the graph induced by W in G. Denote by IDS(G, λ) an independent λ-dominating set of G, i.e., a subset S of non-neighboring nodes such that every node v in G is at distance at most λ of S (namely, $d_G(v, S) \leq \lambda$).

3.2 Description of Algorithm Dom-Span

Algorithm Dom-Span decomposes the node set of G into increasingly denser clusters using a classical merging technique. The algorithm is formally described below. Roughly speaking, at each iteration i, sparse unmerged clusters (kept in the set L) are connected to their neighbors at distance ρ_i using few shortest paths (in the loop of line 8). Next, the dense clusters are merged together (in the loop of line 10). This process is repeated until all the clusters become sparse. Intuitively, connecting sparse neighboring clusters with long shortest paths allows us to obtain a small multiplicative stretch, but it can cause the size of the spanner to increase too much. The general idea of the algorithm is to tune the sequence ρ_i according to each iteration i so that not too many edges are added to the spanner. In fact, as the clusters become dense, the number of clusters decreases and thus we are allowed to connect clusters that lie at a large distance of each other, i.e., the denser the clusters in an iteration i, the larger the distances ρ_i we can choose.

The condition used to evaluate the sparseness of a cluster (line 4) guarantees that the size of the clusters increases exponentially. As a consequence, within a logarithmic number of iterations (in k), all clusters become sparse and all nodes are connected together in the spanner. Using an independent dominating set X on the dense clusters (in line 5) allows us to break the symmetry efficiently and to grow the clusters in parallel.

One key ingredient of our construction is to ensure that the clusters grow sufficiently in each iteration without overlapping. For that purpose, we use an

[3] Note that S itself is not necessarily connected, and the tree T may span also some nodes that do not belong to S.

independent dominating set to pick some independent dense clusters at distance at least 3 from one another (set X of line 5). These independent clusters can then grow in parallel. In fact, using a consistent coloring technique (in the loop of line 6), each cluster determines the cluster in the independent set to which it will be merged. Thus, at least the clusters in the neighborhood of each independent cluster can be merged together in parallel without overlap (the loop of line 10). This process of picking clusters using an independent set allows us to enlarge them sufficiently while preventing new merged clusters from overlapping.

Input: a graph $G = (V, E)$, a sequence $\rho_1, \ldots, \rho_K$ where $K = \lfloor \log k \rfloor + 1$, $\lambda \geq 1$
Output: a spanner H of G

1 $H := (V, \varnothing)$, $\mathcal{C} := \bigcup_{v \in V} (v, \{v\}, (\{v\}, \varnothing))$
2 **for** $i := 1$ *to* K **do**
3 $\quad \mathcal{C}' := \varnothing$, $M := G_{\rho_i}(\mathcal{C})$
4 $\quad L := \{v \in \text{center}(\mathcal{C}) \mid \deg_M(v) \leq n^{1/k} |S_v|\}$ /* sparse clusters */
5 $\quad X := \text{IDS}(M^2[\text{center}(\mathcal{C}) \setminus L], \lambda)$ /* dominating set for the dense clusters */
6 $\quad$ **for** $v \in \text{center}(\mathcal{C})$ **do**
7 $\quad\quad$ **if** $d_M(v, X) \leq 2\lambda$ **then** set $c(v)$ to be its closest node of X in M (breaking ties by identities), **else** $c(v) := \perp$
8 $\quad$ **for** $v, w \in \text{center}(\mathcal{C})$ *such that* v, w *are neighbors in* M *and* $c(v) = \perp$ **do**
9 $\quad\quad$ Add a shortest path in G from S_v to S_w to H
10 $\quad$ **for** $v \in \text{center}(\mathcal{C}) \cap X$ **do**
11 $\quad\quad$ $S := S_v$ and $T := T_v$
12 $\quad\quad$ **for** $w \in \text{center}(\mathcal{C})$ *such that* $c(w) = v$ **do**
13 $\quad\quad\quad$ Compute a shortest path $v = x_0, x_1, \ldots, x_t = w$ in M
14 $\quad\quad\quad$ **for** $j := 1$ *to* t **do**
15 $\quad\quad\quad\quad$ Add a shortest path in G from $S_{x_{j-1}}$ to S_{x_j} to H and to T
16 $\quad\quad\quad\quad$ Add S_{x_j} to S, and add T_{x_j} to T
17 $\quad\quad\quad$ Add (v, S, T) to $\mathcal{C}'$
18 $\quad \mathcal{C} := \mathcal{C}'$

Algorithm 2. Algorithm Dom-Span

3.3 Analysis

Our result is summarized in the following theorem. (Recall the notation (1).) Due to lack of space, the proof is omitted.

Theorem 2. *Algorithm* Dom-Span *is a deterministic distributed algorithm that, for all n-node graph G, integers $k, \lambda \geq 1$, and distance sequence $\rho_1, \ldots, \rho_K$ where $K = \lfloor \log k \rfloor + 1$, computes for G a spanner H of at most $\rho[1, K] \cdot (n^{1+1/k} + n)$ edges. The stretch of H and the time complexity of* Dom-Span *are summarized in the table below.*

In the following table, size and time complexities are stated up to a constant factor. $\text{IDS}(n, \lambda)$ denotes the complexity of computing (distributively) an independent λ-dominating set of an n-node graph. The best currently known bounds are $\text{IDS}(n, 1) = 2^{O(\sqrt{\log n})}$ [37], and $\text{IDS}(n, 2 \log n) = O(\log n)$ (cf. [36]).

stretch	size	time	parameters
$(1+\varepsilon, 8-\varepsilon)$	$\varepsilon^{-1} \cdot n^{4/3}$	$\varepsilon^{-1} + \mathrm{IDS}(n,1)$ $\leq \varepsilon^{-1} + 2^{O(\sqrt{\log n})}$	$\rho_1 = 1, \rho_2 = 8/\varepsilon, k = 3$ $\varepsilon \in (0, 8]$
$(1+\varepsilon, \beta_1)$ $\beta_1 = k^{\log(\log k/\varepsilon)+O(1)}$	$\beta_1 \cdot n^{1+1/k}$	$\beta_1 \cdot \mathrm{IDS}(n,1)$ $\leq \beta_1 \cdot 2^{O(\sqrt{\log n})}$	$\rho_i = (9 \lfloor \log k \rfloor / \varepsilon)^{i-1}$ $\varepsilon \in (0, O(\log k)]$
$(1+\varepsilon, \beta_2)$ $\beta_2 = (\log n/\varepsilon)^{O(\log\log n)}$	$\beta_2 \cdot n$	$\beta_2 \cdot \mathrm{IDS}(n, 2\log n)$ $\leq \beta_2 \cdot \log n$	$\hookrightarrow k = \log n$ $\varepsilon \in (0, O(\log\log n)]$

For $k = 3$, the stretch can be slightly improved as shown in Subsection 3.4. Observe that an independent dominating set ($\lambda = 1$), which is nothing else than a maximal independent set, can be computed in $O(\log n)$ expected time [38], leading to better performances in our algorithm if Las Vegas algorithms are considered.

3.4 An Improved Algorithm for $k = 3$

Finally, we propose a specific construction for $k = 3$, slightly improving the stretch over the general algorithms Local-Span and Dom-Span. The construction, which combines ideas from both algorithms, yields to a $(1+\varepsilon, 6-\varepsilon)$-spanner with $O(\varepsilon^{-1}n^{4/3})$ edges.

Theorem 3. *There is a deterministic distributed algorithm that, for every n-node graph and $\varepsilon \in (0, 6]$, computes a $(1+\varepsilon, 6-\varepsilon)$-spanner of $O(\varepsilon^{-1}n^{4/3})$ edges in $O(\varepsilon^{-1} + \mathrm{IDS}(n,1)) = O(\varepsilon^{-1}) + 2^{O(\sqrt{\log n})}$ time.*

Proof. The proof is based on an ad-hoc construction obtained by merging both algorithms Local-Span and Dom-Span.

Let us denote by D the set of nodes of degree at least $n^{1/3}$, the dense nodes. We first construct a small 2-dominating set X for nodes in D. More formally, we compute a set X such that for every $v \in D$, $d_G(v, X) \leq 2$. This can be done by computing distributively an MIS of $G^2[D]$ in $O(\mathrm{IDS}(n,1))$ time, as done in Algorithm Dom-Span with $\lambda = 1$. We obtain a 2-dominating set X for D with $|X| \leq n^{2/3}$.

From X, we create a partition of D into "clusters", i.e., a connected subgraph $C(x)$ centered in each node x of X and of radius at most 2. This can be done in $O(1)$ time by a vote of each node of D of its closest "dominator" in X, equality being break in a consistence nanner.

For each node v of G select a set $R(v)$ composed of v and of $\min\left\{\deg_G(v), n^{1/3}\right\}$ of its neighbors. This phase is similar to the first phase of Algorithm Local-Span with $k = 3$.

The edges of the spanner H are composed of:

(1) the edges between v and $R(v)$ for each node v of G;
(2) the edges of BFS trees centered at each node $x \in X$ and spanning $C(x)$; and
(3) the edges of the shortest paths computed as follows (the next procedure is similar to the while-loop of Algorithm Local-Span):

for each node $x \in X$, **do** (in parallel):

1. $W(x) := (B(x,\rho) \cap D) \setminus C(x)$, where $\rho = 6/\varepsilon + 2$
2. **while** $W(x) \neq \varnothing$ **do**
   (a) pick the closest $w \in W(x)$ from x;
   (b) add a shortest path in G from x to w; and
   (c) remove from $W(x)$ all nodes v such that $R(v)$ and $R(w)$ intersect.

Time: The time complexity is $O(\mathrm{IDS}(n,1) + \varepsilon^{-1})$, since Phase (3) involves only nodes at distance $O(\varepsilon^{-1})$.

Size: The number of edges for Phase (1) is at most $n^{4/3}$. For Phase (2) this is at most $n - |X|$ since $\{C(x)\}_{x \in X}$ is a partition of the nodes of G. For Phase (3) we observe that the instructions of the while-loop are executed at most $n/n^{1/3} = n^{2/3}$ times since:

- the w's selected at Step 2(a) have pairwise disjoint regions $R(w)$;
- once w is picked, at least all nodes of $R(w)$ are removed from G and cannot be considered any more in $W(x)$, because if $v \in R(w)$ then $R(v) \cap R(w)$ contains at least v since $v \in R(v)$;
- the size of $R(w)$ is $n^{1/3}$ since $w \in W(x) \subseteq D$.

Each path added at Step 2(b) is of length at most ρ, so Phase (3) contributes for at most $|X| \cdot n^{2/3} \cdot \rho \leq \rho \cdot n^{4/3}$ edges. In total, the number of edges of H is at most $(\rho + 1) \cdot n^{4/3} + n = O(\varepsilon^{-1} n^{4/3})$.

Stretch: The stretch analysis is similar to the one of Proposition 7. One consider two distinct nodes u, v in G, and let P be a shortest path from u to v in G. We want to show that $d_H(u,v) \leq (1+\varepsilon) d_G(u,v) + 6 - \varepsilon$, that is H is a $(1+\varepsilon, 6-\varepsilon)$-spanner. We will proceed by induction, prove the result for "small distances", and then assume it holds for all distances $\delta < d_G(u,v)$.

For that, we assume that P is not wholly included in H, since otherwise $d_H(u,v) = d_G(u,v)$ and we are done. Let us first show that:

Claim. If $1 \leq d_G(u,v) \leq \rho - 2$, then $d_H(u,v) \leq d_G(u,v) + 6$.

Proof. Let $u', v' \in P$ respectively be the closest and farthest node from u that are in D. Both nodes exist otherwise P would be composed of only nodes of degree less than $n^{1/3}$ (i.e., not in D), and $P \subseteq H$: contradiction. Note that $d_H(u,u') = d_G(u,u')$ and $d_H(v,v') = d_G(v,v')$. So it suffices to prove that $d_H(u',v') \leq (1+\varepsilon) d' + 6 - \varepsilon$, where $d' = d_G(u',v')$.

Let $x \in X$ such that $u' \in C(x)$. Such x exists, since $u' \in D$. If $v' \in C(x)$, we are done $d_H(u',v') \leq 2$. Note that $d_G(x,v') \leq d_G(x,u') + d_G(u',v') \leq 2 + d'$ that is at most ρ because $d' \leq d_G(u,v) \leq \rho - 2$ by assumption. In other words, $v' \in B(x,\rho)$. It follows that v' is in the set $W(x)$ when initialized at Step 1.

Let w be the node picked at Step 2(a) in the while-loop such that v' is removed from $W(x)$. Let P' be the shortest path added to H from x to w. We have

$R(w) \cap R(v') \neq \varnothing$ ($v' = w$ is possible). Since all the edges from a node z to all its neighbors in $R(z)$ are in H, we have $d_H(w, v') \leq 2$. In other words, there is a route from u' to v' through x and P', and through w to v' thanks to $R(w)$ and $R(v')$. Note that $d_G(x, w) \leq d_G(x, v')$ by the choice of w. Hence, the length of P' is $|P'| \leq d_G(u', w) + 2$ because $d_G(u', x) \leq 2$. So $d_H(u', v') \leq d_H(u', x) + d_H(x, v') \leq 2 + |P'| + 2 \leq d_G(u', v') + 6$, completing the proof of the claim. □

So, if $d_G(u, v) \leq \rho - 2$, then $d_H(u, v) \leq d_G(u, v) + 6 \leq (1 + \varepsilon)d_G(u, v) + 6 - \varepsilon$ since $d_G(u, v) \geq 1$.

Assume now that $d_G(u, v) > \rho - 2$, and let $z \in P$ be such that $d_G(u, z) = \rho - r$. By definition of ρ and the choice of $\varepsilon \in (0, 6]$, $\rho - 2 \geq 1$. Therefore $u \neq z$ and Claim 3.4 applies: $d_H(u, z) \leq \rho - 2 + 6$. Observe that $\varepsilon(\rho - 2) \geq 6$, and thus $d_H(u, z) \leq \rho - 2 + \varepsilon(\rho - 2) = (1 + \varepsilon)(\rho - 2)$.

By induction hypothesis on the distance between z and v, which is $< d_G(u, v)$, we get: $d_H(z, v) \leq (1+\varepsilon)d_G(z, v)+6-\varepsilon = (1+\varepsilon)(d_G(u, v) - d_G(u, z))+6-\varepsilon$. If follows that $d_H(u, v) \leq d_H(u, z) + d_H(z, v) \leq (1+\varepsilon)(\rho-2)+(1+\varepsilon)(d_G(u, v) - (\rho - 2)) + 6 - \varepsilon = (1 + \varepsilon)d_G(u, v) + 6 - \varepsilon$. This completes the stretch analysis, and the proof of Theorem 3. □

References

1. Jacquet, P., Viennot, L.: Remote spanners: what to know beyond neighbors. In: 23^{rd} IEEE International Parallel & Distributed Processing Symposium (IPDPS), IEEE Computer Society Press, Los Alamitos (2009)
2. Adjih, C., Laouiti, A., Muhlethaler, P., Qayyum, A., Viennot, L.: The optimised routing protocol for mobile ad-hoc networks: protocol specification. Technical Report 5145, INRIA (March 2003)
3. Peleg, D., Ullman, J.D.: An optimal synchronizer for the hypercube. In: 6^{th} Annual ACM Symposium on Principles of Distributed Computing (PODC), pp. 77–85. ACM Press, New York (1987)
4. Peleg, D., Schäffer, A.A.: Graph spanners. Journal of Graph Theory 13(1), 99–116 (1989)
5. Awerbuch, B.: Complexity of network synchronization. Journal of the ACM 32(4), 804–823 (1985)
6. Peleg, D., Ullman, J.D.: An optimal synchornizer for the hypercube. SIAM Journal on Computing 18(4), 740–747 (1989)
7. Cohen, E.: Fast algorithms for constructing t-spanners and paths with stretch t. SIAM Journal on Computing 28(1), 210–236 (1998)
8. Elkin, M., Zhang, J.: Efficient algorithms for constructing $(1+\epsilon, \beta)$-spanners in the distributed and streaming models. Distributed Computing 18(5), 375–385 (2006)
9. Baswana, S., Goyal, V., Sen, S.: All-pairs nearly 2-approximate shortest-paths in $O(n^2 \text{polylog} n)$ time. In: Diekert, V., Durand, B. (eds.) STACS 2005. LNCS, vol. 3404, pp. 666–679. Springer, Heidelberg (2005)
10. Baswana, S., Kavitha, T.: Faster algorithms for approximate distance oracles and all-pairs small stretch paths. In: 47^{th} Annual IEEE Symposium on Foundations of Computer Science (FOCS), pp. 591–602. IEEE Computer Society Press, Los Alamitos (2006)

11. Thorup, M., Zwick, U.: Approximate distance oracles. Journal of the ACM 52(1), 1–24 (2005)
12. Roditty, L., Thorup, M., Zwick, U.: Deterministic constructions of approximate distance oracles and spanners. In: Caires, L., Italiano, G.F., Monteiro, L., Palamidessi, C., Yung, M. (eds.) ICALP 2005. LNCS, vol. 3580, pp. 261–272. Springer, Heidelberg (2005)
13. Narasimhan, G., Smid, M.: Geometric Spanner Networks. Cambridge University Press, Cambridge (2007)
14. Spielman, D.A., Teng, S.H.: Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems. In: 36^{th} Annual ACM Symposium on Theory of Computing (STOC), pp. 81–90. ACM Press, New York (2004)
15. Dor, D., Halperin, S., Zwick, U.: All-pairs almost shortest paths. SIAM Journal on Computing 29(5), 1740–1759 (2000)
16. Thorup, M., Zwick, U.: Spanners and emulators with sublinear distance errors. In: 17^{th} Symposium on Discrete Algorithms (SODA), January 2006, pp. 802–809. ACM/ SIAM (2006)
17. Bollobás, B., Coppersmith, D., Elkin, M.: Sparse distance preservers and additive spanners. SIAM Journal of Discrete Mathematics 19(4), 1029–1055 (2006)
18. Chechik, S., Langberg, M., Peleg, D., Roditty, L.: Fault-tolerant spanners for general graphs. In: 41^{st} Annual ACM Symposium on Theory of Computing (STOC). ACM Press, New York (2009)
19. Roditty, L., Thorup, M., Zwick, U.: Roundtrip spanners and roundtrip routing in directed graphs. ACM Transactions on Algorithms 3(4), Article 29 (2008)
20. Pettie, S.: Low distortion spanners. In: Arge, L., Cachin, C., Jurdziński, T., Tarlecki, A. (eds.) ICALP 2007. LNCS, vol. 4596, pp. 78–89. Springer, Heidelberg (2007)
21. Althöfer, I., Das, G., Dobkin, D., Joseph, D.A., Soares, J.: On sparse spanners of weighted graphs. Discrete & Computational Geometry 9(1), 81–100 (1993)
22. Erdös, P.: Extremal problems in graph theory, pp. 29–36. Publ. House Cszechoslovak Acad. Sci., Prague (1964)
23. Erdös, P., Simonovits, M.: Compactness results in extremal graph theory. Combinatorica 2(3), 275–288 (1982)
24. Aingworth, D., Chekuri, C., Indyk, P., Motwani, R.: Fast estimation of diameter and shortest paths (without matrix multiplication). SIAM Journal on Computing 28(4), 1167–1181 (1999)
25. Baswana, S., Kavitha, T., Mehlhorn, K., Pettie, S.: New constructions of (α, β)-spanners and purely additive spanners. In: 16^{th} Symposium on Discrete Algorithms (SODA), pp. 672–681. ACM/ SIAM (2005)
26. Woodruff, D.P.: Lower bounds for additive spanners, emulators, and more. In: 47^{th} Annual IEEE Symposium on Foundations of Computer Science (FOCS), pp. 389–398. IEEE Computer Society Press, Los Alamitos (2006)
27. Elkin, M., Peleg, D.: $(1 + \epsilon, \beta)$-spanner constructions for general graphs. SIAM Journal on Computing 33(3), 608–631 (2004)
28. Pettie, S.: Distributed algorithms for ultrasparse spanners and linear size skeletons. In: 27^{th} Annual ACM Symposium on Principles of Distributed Computing (PODC), pp. 253–262. ACM Press, New York (2008)
29. Baswana, S., Sen, S.: A simple linear time algorithm for computing a $(2k-1)$-spanner of $O(n^{1+1/k})$ size in weighted graphs. In: Baeten, J.C.M., Lenstra, J.K., Parrow, J., Woeginger, G.J. (eds.) ICALP 2003. LNCS, vol. 2719, pp. 384–396. Springer, Heidelberg (2003)

30. Derbel, B., Gavoille, C., Peleg, D., Viennot, L.: On the locality of distributed sparse spanner construction. In: 27^{th} Annual ACM Symposium on Principles of Distributed Computing (PODC), pp. 273–282. ACM Press, New York (2008)
31. Elkin, M.: Computing almost shortest paths. ACM Transactions on Algorithms 1(2), 283–323 (2005)
32. Elkin, M., Zhang, J.: 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), pp. 160–168. ACM Press, New York (2004)
33. Elkin, M., Peleg, D.: $(1 + \epsilon, \beta)$-spanner constructions for general graphs. In: 33^{rd} Annual ACM Symposium on Theory of Computing (STOC), pp. 173–182. ACM Press, New York (2001)
34. Kuhn, F., Moscibroda, T., Wattenhofer, R.: What cannot be computed locally! In: 23^{rd} Annual ACM Symposium on Principles of Distributed Computing (PODC), pp. 300–309. ACM Press, New York (2004)
35. Linial, N.: Locality in distributed graphs algorithms. SIAM Journal on Computing 21(1), 193–201 (1992)
36. Peleg, D.: Proximity-preserving labeling schemes. Journal of Graph Theory 33(3), 167–176 (2000)
37. Panconesi, A., Srinivasan, A.: On the complexity of distributed network decomposition. Journal of Algorithms 20(2), 356–374 (1996)
38. Luby, M.: A simple parallel algorithm for the maximal independent set problem. SIAM Journal on Computing 15(4), 1036–1053 (1986)