SIAM J. COMPUT.
Vol. 33, No. 3, pp. 608–631

$(1+\epsilon,\beta)$-SPANNER CONSTRUCTIONS FOR GENERAL GRAPHS*

MICHAEL ELKIN† AND DAVID PELEG‡

Abstract. An (α,β)-*spanner* of a graph G is a subgraph H such that $dist_H(u,w) \leq \alpha \cdot distt_G(u,w)+\beta$ for every pair of vertices u,w, where $dist_{G'}(u,w)$ denotes the distance between two vertices u and v in G'. It is known that every graph G has a polynomially constructible $(2\kappa-1,0)$-spanner (also known as *multiplicative* $(2\kappa-1)$-spanner) of size $O(n^{1+1/\kappa})$ for every integer $\kappa \geq 1$, and a polynomially constructible $(1,2)$-spanner (also known as *additive* 2-spanner) of size $\tilde{O}(n^{3/2})$. This paper explores hybrid spanner constructions (involving both multiplicative and additive factors) for general graphs and shows that the multiplicative factor can be made *arbitrarily close to* 1 while keeping the spanner size arbitrarily close to $O(n)$, at the cost of allowing the additive term to be a sufficiently large constant. More formally, we show that for any constant $\epsilon,\lambda > 0$ there exists a constant $\beta = \beta(\epsilon,\lambda)$ such that for every n-vertex graph G there is an efficiently constructible $(1+\epsilon,\beta)$-spanner of size $O(n^{1+\lambda})$.

Key words. spanners, graph algorithms, graph partitions

AMS subject classifications. 05C85, 05C12, 68R10

DOI. 10.1137/S0097539701393384

1. Introduction.

1.1. Motivation and previous results.

Spanners for general graphs were introduced in [15] as a tool for constructing synchronizers, and were later studied in various contexts. Generally speaking, spanners appear to be the underlying graph structure in a number of constructions in distributed systems and communication networks (cf. [13]). Spanners have turned out to be relevant also in other contexts. For instance, in the area of robotics and computational geometry, spanners have been considered in the Euclidean setting, assuming that the vertices of the graph are points in space, and the edges are line segments connecting pairs of points (cf. [5, 6, 1] and the references therein).

Intuitively, spanners can be thought of as a generalization of the concept of a spanning tree, allowing the spanning subgraph to have cycles, but aiming towards maintaining the locality properties of the network. These locality properties revolve around the notion of *stretch*, namely, the (worst) multiplicative factor by which distances increase in the network as a result of using the spanner edges alone and ignoring non-spanner edges. Formally, given an unweighted graph $G=(V,E)$, we say that the subgraph $H=(V,E')$ (where $E' \subseteq E$) is an α-*spanner* of G if $dist_H(u,w) \leq \alpha \cdot dist_G(u,w)$ for every $u,w \in V$, where $dist_{G'}(u,v)$ denotes the distance between two vertices u and v in G', namely, the minimum length of a path in G' connecting them.

There exists a well-understood tradeoff between the size of a spanner (namely, the number of edges it uses) and its stretch [14, 1, 4]. Generally speaking, for an integer parameter κ, a stretch of $O(\kappa)$ can be guaranteed by a spanner using $O(n^{1+1/\kappa})$ edges. The best known bound for the stretch is $2\kappa-1$ [1]. This bound is very close to the

*Received by the editors August 5, 2001; accepted for publication (in revised form) November 12, 2003; published electronically March 30, 2004.
http://www.siam.org/journals/sicomp/33-3/39338.html

†Department of Computer Science, Yale University, New Haven, CT 06520 (elkin@cs.yale.edu). This author's work was done at the Weizmann Institute of Science, Rehovot, Israel.

‡Department of Computer Science and Applied Mathematics, Weizmann Institute of Science, Rehovot, 76100 Israel (peleg@wisdom.weizmann.ac.il). This author's work was supported in part by grants from the Israel Science Foundation and the Israel Ministry of Science and Art.

best possible in view of the lower bounds shown in [14, 1, 4], by which for every odd $\kappa \geq 3$ there exist (infinitely many) n-vertex graphs $G = (V, E)$ for which every $(\kappa - 2)$-spanner requires $\Omega(n^{1+4/3\kappa})$ edges. In fact, for 3-, 5-, and 9-spanners, even better lower bounds are known [16].

However, it is not clear a priori that the stretch must be expressed as a *multiplicative* factor. An alternative notion of *additive graph spanners* was introduced in [11, 12]. A subgraph H is an *additive β-spanner* of G if $dist_H(u, w) \leq dist_G(u, w) + \beta$ for every $u, w \in V$. While the results of [11, 12] concerned only special graph classes, like pyramids, hypercubes and multidimensional grids of degree bounded by 4, a result demonstrating the potential usefulness of this notion for general graphs was presented in [7], where it was shown that for every graph G there exists an additive 2-spanner with $\tilde{O}(n^{3/2})$ edges. Again, this is the best possible up to polylogarithmic factors given the aforementioned lower bounds. Unfortunately, this result has so far resisted attempts of extending it to values of κ greater than 2.

In this paper, we study the somewhat more general and unifying concept of (α, β)-*spanners*, also introduced in [11]. A subgraph H is an (α, β)-*spanner* of G if $dist_H(u, w) \leq \alpha \cdot dist_G(u, w) + \beta$ for every $u, w \in V$. Cast in this terminology, the above mentioned results imply that $O(n^{1+1/\kappa})$ edges suffice to construct a $(2\kappa - 1, 0)$-spanner, for any integer $\kappa \geq 1$, and that $\tilde{O}(n^{3/2})$ edges suffice to construct a $(1, 2)$-spanner.

However, intuitively it seems that a tighter bound on the behavior of spanners may be obtained. In particular, it seems plausible that the multiplicative factor of $2\kappa - 1$ is not entirely unavoidable, as it might stem in part due to a "hidden" additive term affecting mainly the stretching behavior of the spanner with respect to *nearby* vertex pairs.

The current paper not only confirms this intuition, but also shows that the multiplicative stretch can be made *arbitrarily close to* 1 by allowing the additive term to be a sufficiently large *constant*.

1.2. Our results. Our main construction establishes the existence and efficient constructibility of $(1 + \epsilon, \beta)$-spanners of size $O(\beta n^{1+1/\kappa})$ for every n-vertex graph G, where $\beta = \beta(\kappa, \epsilon)$ is constant whenever κ and ϵ are. We stress that in our construction, both the stretch and the size of the spanner can be made arbitrarily small *simultaneously*, where the tradeoff is between their values on the one hand and the value of the additive term on the other.

Note that the existence of spanners with properties as described above implies that for any constant $\epsilon, \lambda > 0$ and graph G, there exists a spanning subgraph H with $O(n^{1+\lambda})$ edges which behaves like a $(1 + \epsilon)$-spanner for "sufficiently distant" pairs of vertices. More formally, for any constant $\epsilon, \lambda > 0$ there exists a constant $\beta(\epsilon, \lambda)$ such that for any n-vertex graph $G = (V, E)$ there exists an efficiently constructible spanning subgraph H with $O(n^{1+\lambda})$ edges such that $dist_H(u, w) \leq (1 + \epsilon) dist_G(u, w)$ for every pair of vertices $u, w \in V$ such that $dist_G(u, w) \geq \beta(\epsilon, \lambda)$.

We remark that this result is optimal in the sense that the statement becomes false if either ϵ or λ is set to zero. Specifically, taking $\epsilon = 0$ yields a false statement since for any $0 < \lambda < 1$ there exists an infinite graph family $\mathcal{H}(\lambda)$ such that every n-vertex graph $G \in \mathcal{H}(\lambda)$ has $\Omega(n^{1+\lambda})$ edges and for any subgraph H of G there exists a pair of nodes u, w in the graph such that $d_G(u, w) = \Omega(n^{1/2-\lambda/2})$ and $d_H(u, w) > d_G(u, w)$. Analogously, setting λ to zero falsifies the statement because for any $0 < \epsilon < 1$ and $\beta(\epsilon) \geq 1$ there exists an infinite graph family $\hat{\mathcal{H}}(\epsilon, \beta)$ such that any n-vertex graph $G \in \hat{\mathcal{H}}(\epsilon, \beta)$ has $n^{1+\Omega(1/\beta(\epsilon))}$ edges (i.e., significantly more than $O(n) = O(n^{1+\lambda})$)

and for any subgraph H of G there exists a pair of nodes u, w in the graph such that $d_G(u,w) \geq \beta(\epsilon)$ and $d_H(u,w) \geq 2\beta(\epsilon)$ [3].

Furthermore, our construction enables us to obtain a multiplicative stretch that is even smaller than $1+\epsilon$ for constant $\epsilon > 0$, i.e., $1+1/\operatorname{polylog} n$ or even $1+2^{-\log n/\log^{(b)} n}$ (where $\log^{(b)}$ denotes the b-iterated log function), while still keeping the size of the spanner equal to $O(n^{1+\lambda})$ for arbitrarily small $\lambda > 0$, at the cost of increasing the additive term to $\operatorname{polylog} n$ or $\exp(O(\log n/\log^{(b)} n))$, respectively.

A straightforward implementation of our construction requires $O(|E| \cdot n)$ time, but we show that our construction can be implemented even faster, and establish a tradeoff between the time complexity of the construction algorithm and the additive term. Specifically, we show that a $(1+\epsilon, \beta')$-spanner of size $O(\beta' n^{1+1/\kappa})$ can be constructed in $\tilde{O}(n^{2+\mu})$ time for any n-vertex graph G, where the additive term $\beta' = \beta'(\kappa, \epsilon, \mu)$ is constant whenever κ, ϵ, and μ are.

The additive terms involved in our constructions are roughly $\beta(\kappa,\epsilon) = \kappa^{\log\log\kappa - \log\epsilon}$ and $\beta'(\kappa,\epsilon,\mu) = \max\{\beta(\kappa,\epsilon), \kappa^{-\log\mu}\}$. Although optimizing these additive terms is not in the focus of the current paper, we remark that β and β' are always $O((\log n)^{\log^{(3)} n})$, because $\kappa = O(\log n)$. Indeed, for $\kappa = \Omega(\log n)$ the size of the spanner becomes almost linear (i.e., $O(n \cdot (\log n)^{\log^{(3)} n})$).

Finally, analysis of our construction for specific small values of κ enables us to derive some secondary results. First, a construction of an additive 2-spanner of size $O(n^{3/2})$ can be derived. This is tight up to a constant factor due to lower bound of [16], and improves upon the construction of [7] by a logarithmic factor. However, the running time of our construction is $O(n^{5/2})$ instead of $\tilde{O}(n^2)$ of the construction of [7]. Additionally, a construction of a $(1+\epsilon, 4)$-spanner of size $O(\epsilon^{-1} n^{4/3})$ can be derived. This improves the previously known construction of a multiplicative 5-spanner of size $O(n^{4/3})$. The details of the analysis of our algorithm for small values of κ were presented in the preliminary versions of this paper [9, 10] and are omitted here.

Since the appearance of a preliminary version of this paper [10], a more time-efficient construction of $(1+\epsilon, \beta)$-spanners for general graphs was devised in [8]. Specifically, it is shown therein that $(1+\epsilon, \beta'')$-spanners of size $O(\beta'' n^{1+1/\kappa})$ can be constructed in $O(|E| n^{\mu})$ time, where the additive term $\beta'' = \beta''(\kappa, \epsilon, \mu)$ is constant whenever κ, ϵ, and μ are. The improved running time of the construction of [8] makes it possible to use $(1+\epsilon, \beta)$-spanners as a building block of the most efficient currently known algorithm for computing almost shortest paths with constant almost additive error from s, $1 \ll s \ll n$, sources. However, the additive term β'' in the construction of [8] is asymptotically greater than the additive term β' in the present construction.

2. The construction algorithm. In this section we present the polynomial time algorithm that for any constant $0 < \epsilon < 1$ and constant integer $\kappa \geq 2$, and for any n-vertex graph G, constructs a $(1+\epsilon, \beta)$-spanner with $O(\beta n^{1+1/\kappa})$ edges, where $\beta = \beta(\epsilon, \kappa)$ is a constant (independent of n and of any other parameter of the graph).

We start with some necessary definitions. We refer to the vertex and edge sets of a subgraph G' by $V(G')$ and $E(G')$, respectively. For any subgraph $G' = (V', E')$ and any integer $\ell \geq 0$ and vertex $v \in V'$, we denote the ℓ-neighborhood of v in G' by $\Gamma_\ell^{G'}(v) = \{u \mid dist_{G'}(v,u) \leq \ell\}$. For any subset of vertices $U \subseteq V'$, denote the set of neighbors of U in G' by $\Gamma^{G'}(U) = \{z \mid \exists u \in U \text{ such that } (u,z) \in E'\}$. We also denote $\Gamma_\ell(v) = \Gamma_\ell^G(v)$ and $\Gamma(U) = \Gamma^G(U)$.

Input: Graph $G = (V, E)$.
Output: Spanned partition $\mathcal{G}$, subgraph H.

1. $U \leftarrow V$; $\mathcal{G} \leftarrow \emptyset$; $\mathcal{S}' \leftarrow \emptyset$; $E(H) \leftarrow \emptyset$;
2. **While** $U \neq \emptyset$ **do:**
   (a) Pick an arbitrary vertex $v \in U$;
   (b) $S \leftarrow \{v\}$;
   (c) **While** $|S \cup \Gamma(S) \cap U| \geq n^{1/\kappa}|S|$ **do:**
      • $S \leftarrow S \cup \Gamma(S) \cap U$; /* the cluster
   (d) $\hat{S} \leftarrow S \cup \Gamma(S) \cap U$; /* the shell
   (e) Form a BFS spanning tree T for (v, S);
   (f) $\mathcal{S}' \leftarrow \mathcal{S}' \cup \{(v, \hat{S})\}$; $\mathcal{G} \leftarrow \mathcal{G} \cup \{(v, S, T)\}$; $U \leftarrow U \setminus S$;
3. **For every** $(v, \hat{S}) \in \mathcal{S}'$ **do:**
   (a) Create a BFS spanning tree T' rooted at v for $\hat{S}$ (namely, a tree spanning $\hat{S}$ and yielding shortest paths to v in the induced subgraph $G(\hat{S})$);
   (b) $E(H) \leftarrow E(H) \cup E(T')$;
4. Return$(\mathcal{G}, H)$;

FIG. 1. *Procedure* DOWN_PART.

For a subset of vertices $W \subseteq V$, let $G(W)$ denote the subgraph of G induced by W. With a slight abuse of notation, for a subset $E' \subseteq E$ of edges, let $G(E')$ denote the graph $(V(E'), E')$, where $V(E') = \{v \in V \mid \exists e \in E' s.t.\ v \in e\}$. A subset of vertices $C \subseteq V$ is called a *cluster* if its induced subgraph $G(C)$ is connected. A triple (v, S, T) is called a *spanned cluster* if $v \in S$, and T is a connected spanning tree of S. (Note that S itself is not necessarily connected, and the tree T may span also some vertices that do not belong to S.) The *radius* of a spanned cluster (v, S, T) is $rad(v, S, T) = \max\{dist_{G(T)}(v, u) \mid u \in S\}$, and the *diameter* is $diam(v, S, T) = \max\{dist_{G(T)}(u, w) \mid u, w \in S\}$. Since in a spanned cluster (v, S, T), the tree T is necessarily connected, the radius and the diameter of a spanned cluster are always finite. The spanned clusters (v_i, S_i, T_i) and (v_j, S_j, T_j) are *disjoint* if $S_i \cap S_j = \emptyset$. A set of disjoint spanned clusters $\{(v_i, S_i, T_i)\}$ is called a *spanned partition*. For any pair of vertex sets $U_1, U_2 \subseteq V$ denote $dist_G(U_1, U_2) = \min\{dist_G(u_1, u_2) \mid u_1 \in U_1, u_2 \in U_2\}$. For a spanned partition $\mathcal{U}$, a cluster S, and an integer ℓ, denote

$$\Gamma^{\mathcal{U}}_{\ell}(S) \;=\; \{(v_i, S_i, T_i) \in \mathcal{U} \mid dist_G(S_i, S) \leq \ell\}\ .$$

2.1. The initial partitioning procedure. Our algorithm starts by invoking Procedure DOWN_PART, described in Figure 1. This procedure is a variant of the procedures for constructing partitions due to [14, 7]. The procedure creates a spanned partition $\mathcal{G}$, which we also call the *ground partition*, since the algorithm uses it as a basis for constructing other partitions. (This partition satisfies that $\bigcup_i S_i = V$; the partitions constructed later on may be *partial*, i.e., they will not necessarily have this property.) For each cluster that the procedure creates, it also builds a "shell" consisting of the cluster and one external layer, in a way similar to the partitioning algorithm of [2]. The procedure also creates a subgraph H, which is a subset, and in some sense a *core*, of the spanner created by the algorithm. This subgraph H consists of the union of the breadth first search (BFS) trees of all the shells created by the procedure. (Given a cluster S centered at a vertex v in the graph G, a *BFS tree* for (v, S) is a tree spanning S which yields shortest paths to v in the induced subgraph $G(S)$.)

Let us next establish some basic properties concerning the output of Procedure DOWN_PART. Given a spanned partition $\mathcal{S} = \{(v_j, S_j, T_j)\}$, denote by $\mathcal{A}_i(\mathcal{S})$ the subset of spanned clusters of $\mathcal{S}$ with radius i,

$$\mathcal{A}_i(\mathcal{S}) \;=\; \{(v,S,T) \in \mathcal{S} \mid rad(v,S,T) = i\}.$$

For any spanned partition $\mathcal{S}$ denote the minimum cluster size by $\check{S}(\mathcal{S}) = \min_{S\in\mathcal{S}} |S|$.

LEMMA 2.1. *Let $\mathcal{G}$ be the ground partition returned by Procedure* DOWN_PART. *Then $\check{S}(\mathcal{A}_j(\mathcal{G})) \geq n^{j/\kappa}$ for any integer $0 \leq j \leq \kappa - 1$.*

Proof. Consider a spanned cluster $(v,S,T) \in \mathcal{A}_j(\mathcal{G})$. By definition of $\mathcal{A}_j(\mathcal{G})$, $rad(v,S,T) = j$. Each cluster constructed by Procedure DOWN_PART starts with radius zero, and each iteration of the internal while loop (step 2(c)) of the procedure increases the radius of the cluster by at most 1. Hence the cluster (v,S,T) was involved in the loop for at least j iterations. In each iteration its size grew by a factor of at least $n^{1/\kappa}$. The lemma follows.

Next, we argue that the subgraph H returned by Procedure DOWN_PART is sparse.

LEMMA 2.2. $|E(H)| \;=\; O(n^{1+1/\kappa})$.

Proof. For every cluster $S \in \mathcal{S}$, the shell consists of $\hat{S} = S \cup \Gamma(S) \cap U$ at the time of insertion, so by the condition of step 2(c) for selecting the cluster, $|\hat{S}| < n^{1/\kappa} \cdot |S|$. Hence the number of edges inserted into H in step 3 of Procedure DOWN_PART is bounded by

$$\sum_{S\in\mathcal{S}} |S| n^{1/\kappa} \;=\; n^{1/\kappa} \sum_{S\in\mathcal{S}} |S| \;=\; O(n^{1+1/\kappa}),$$

since the clusters S are disjoint. ∎

An additional important property of the ground partition $\mathcal{G}$ and the subgraph H returned by the invocation DOWN_PART(G) is that for any pair of neighboring clusters of $\mathcal{G}$ and for any edge e between these clusters, there is an edge between these clusters in H that is incident to one of the endpoints of e. More specifically, we introduce the following key definition.

DEFINITION 2.3. *A spanned partition $\mathcal{S}$ of a graph $G = (V,E)$ is said to be* adjacency-preserving *with respect to a subgraph H such that $E(H) \subseteq E$ if it satisfies the following two properties.*

(P1) *For any spanned cluster $(v,S,T) \in \mathcal{S}$, there exists a BFS spanning tree T' for (v,S) such that $E(T') \subseteq E(H)$.*

(P2) *For any pair of neighboring clusters $(v_1,S_1,T_1), (v_2,S_2,T_2) \in \mathcal{S}$ (i.e., such that $dist_G(S_1,S_2) = 1$) and for any edge $e = (u_1,u_2) \in E$ such that $u_1 \in S_1$ and $u_2 \in S_2$, there exists either a node $u_2' \in S_2$ such that $(u_1,u_2') \in E(H)$ or a node $u_1' \in S_1$ such that $(u_1',u_2) \in E(H)$. In the former (resp., latter) case, the edge e is said to be* spanned through *the cluster S_2 (resp., S_1).*

Note that in the definition above, if the edge (u_1,u_2) is spanned through the cluster S_i, for $i \in \{1,2\}$, then $dist_H(u_1,u_2) \leq diam(S_i) + 1$.

LEMMA 2.4. *Let $(\mathcal{G},H)$ be the pair returned by Procedure* DOWN_PART(G). *Then the spanned partition $\mathcal{G}$ is adjacency-preserving with respect to H.*

Proof. To prove that the pair $(\mathcal{G},H)$ satisfies property (P1) of Definition 2.3, consider some spanned cluster (v,S,T). Note that at step 3(a) of Procedure DOWN_PART, the BFS spanning tree T' of its shell $\hat{S}$ rooted at v was inserted into the subgraph H. Note that such a tree is, in particular, a BFS spanning tree for S rooted at v (although not necessarily the same as T), completing the proof of property (P1).

To prove that $(\mathcal{G}, H)$ satisfies property (P2) as well, consider some pair of neighboring spanned clusters (v_1, S_1, T_1) and (v_2, S_2, T_2), and some edge $e = (u_1, u_2)$ between them, such that $u_1 \in S_1$ and $u_2 \in S_2$. Assume, without loss of generality, that the cluster S_1 was created before the cluster S_2. Consider the iteration of the main loop of Procedure DOWN_PART on which S_1 was created. Denote by U' the set U at the beginning of this iteration. Note that at this stage all the nodes of S_1 and S_2 were still uncovered, i.e., $S_1, S_2 \subseteq U'$, and thus in particular $u_1, u_2 \in U'$. It follows that e is in $G' = G(U')$, the subgraph of G induced by U', and hence $u_2 \in \Gamma^{G'}(S_1) \subseteq \hat{S}_1$.

Denote the radius of S_1 by $\rho = rad(v_1, S_1, T_1)$. By step 2(c) of Procedure DOWN_PART, $\Gamma^{G'}_{\rho}(v_1) = S_1$ and $\Gamma^{G'}_{\rho+1}(v_1) = \hat{S}_1$. Recalling that $u_2 \in \hat{S}_1 \setminus S_1$, we conclude that $dist_{G'}(v_1, u_2) = \rho + 1$.

The spanning tree T' constructed for $\hat{S}_1$ at step 3(a) of Procedure DOWN_PART spans $\hat{S}_1$, and thus $u_2 \in V(T')$. Since T' is a BFS spanning tree with respect to v_1 in the induced subgraph $G'(\hat{S}_1)$, and since $dist_{G'}(v_1, u_2) = \rho + 1$, it follows that the parent z_1 of u_2 in T' satisfies $dist_{G'}(v_1, z_1) = \rho$, and thus $z_1 \in \Gamma^{G'}_{\rho}(v_1) = S_1$, as required. ∎

2.2. The superclustering procedure. We next proceed to describing the *superclustering* procedure SC. The procedure receives a spanned partition $\mathcal{C}$, identifies its "dense" sets of clusters, and merges them into *superclusters*. By a dense set of clusters we mean a set containing "fairly many" clusters, which are "close enough" to each other. These qualitative notions are quantified by two additional parameters. The *size* parameter σ of Procedure SC specifies the minimum number of clusters close to a given cluster S that justifies creating a supercluster around S that will contain them, whereas the *distance* parameter δ specifies when two clusters are considered to be close. Procedure SC returns a spanned partition $\mathcal{C}'$ which contains the superclusters created and a subgraph H that will later be added into the spanner. When Procedure SC finishes creating superclusters, it remains with a collection $\mathcal{R}$ of original clusters of the input partition that were left untouched. Procedure SC then finds shortest paths between close pairs in the collection $\mathcal{R}$ and inserts these paths as well into the output subgraph H. The size parameter σ controls also the number of pairs of close $\mathcal{R}$ clusters and therefore the size of the subgraph H. The procedure is described in Figure 2.

Note that in step 2(b)ii, while augmenting the set of edges T' associated with a supercluster S' by adding the edges of the shortest path P_i, we do not insert the vertices of this path into the vertex set S' associated with this supercluster. The reason is that these vertices are clustered in other clusters of the ground partition and they may be superclustered separately. In other words, the same vertex may participate in the shortest path connecting two clusters of one supercluster, and at the same time be clustered in another supercluster. Hence T' is not necessarily contained in $E(G(S'))$.

Intuitively, the following lemma states that Procedure SC does not destroy the adjacency-preservation property.

LEMMA 2.5. *Let the triple $(\mathcal{C}', H', \mathcal{R})$ be the output of the invocation* SC$(G, \mathcal{C}, \sigma, \delta)$ *for a graph $G = (V, E)$, its spanned partition $\mathcal{C}$ and some arbitrary σ and δ. Suppose that the spanned partition $\mathcal{C}$ is adjacency-preserving with respect to some subgraph H of G such that $E(H) \subseteq E$. Then the output spanned partitions $\mathcal{C}'$ and $\mathcal{R}$ are adjacency-preserving with respect to $H \cup H'$.*

Proof. Note first that the adjacency-preservation property is monotone in both parameters (i.e., if a spanned partition $\tilde{\mathcal{S}}$ is adjacency-preserving with respect to

Input: Graph $G = (V, E)$, spanned partition $\mathcal{C} = \{(v_i, S_i, T_i)\}$ of G, integers σ, $\delta \geq 1$.
Output: Spanned partition $\mathcal{C}' = \{(v'_i, S'_i, T'_i)\}$ of G, subgraph H', collection $\mathcal{R}$ of unmerged clusters.

1. $E(H') \leftarrow \emptyset$; $\mathcal{U} \leftarrow \mathcal{C}$; $\mathcal{C}' \leftarrow \emptyset$;
2. **while** there exists a spanned cluster $(v, S, T) \in \mathcal{U}$ such that $|\Gamma^{\mathcal{U}}_{\delta}(S) \cap \mathcal{U}| \geq \sigma$ **do:**
   (a) $S' \leftarrow \bigcup_{(v_i, S_i, T_i) \in \Gamma^{\mathcal{U}}_{\delta}(S)} S_i$;
   (b) i $T' \leftarrow T$;
   ii **For** every S_i such that $(v_i, S_i, T_i) \in \Gamma_{\mathcal{U}}(S)$ **do:**
   A. Compute the shortest path P_i in G between the clusters S and S_i;
   B. $E(T') \leftarrow E(T') \cup E(P_i) \cup E(T_i)$;
   (c) $\mathcal{C}' \leftarrow \mathcal{C}' \cup \{(v, S', T')\}$;
   (d) $\mathcal{U} \leftarrow \mathcal{U} \setminus \Gamma^{\mathcal{U}}_{\delta}(S)$;
   (e) $E(H') \leftarrow E(H') \cup E(T')$;
3. $\mathcal{R} \leftarrow \mathcal{U}$; /* Remaining (unmerged) clusters */
4. **For** every pair of spanned clusters $(v_i, S_i, T_i), (v_j, S_j, T_j) \in \mathcal{R}$ such that $dist_G(S_i, S_j) \leq \delta$ **do:**
   (a) Compute the shortest path P_{ij} between S_i and S_j in G;
   (b) $E(H') \leftarrow E(H') \cup E(P_{ij})$;
5. Return$(\mathcal{C}', H', \mathcal{R})$;

FIG. 2. *Procedure* SC.

some subgraph $\tilde{H}$, then this spanned partition is adjacency-preserving with respect to any subgraph $\bar{H}$ such that $E(\tilde{H}) \subseteq E(\bar{H}) \subseteq E$). Also, if $\bar{\mathcal{C}} \subseteq \tilde{\mathcal{C}}$, and $\tilde{\mathcal{C}}$ is adjacency-preserving with respect to some subgraph $\tilde{H}$, then the spanned partition $\bar{\mathcal{C}}$ is adjacency-preserving with respect to $\tilde{H}$ as well.

Now the statement of the lemma regarding the spanned partition $\mathcal{R}$ follows from the observation that $\mathcal{R} \subseteq \mathcal{C}$, and from the assumption of the lemma that $\mathcal{C}$ is adjacency-preserving with respect to H.

It remains to establish the lemma for $\mathcal{C}'$. Observe that the pair $(\mathcal{C}', H \cup H')$ satisfies property (P1) of Definition 2.3, by the same assumption of the lemma, and by step 2(e) of Procedure SC. To prove that the pair $(\mathcal{C}', H \cup H')$ satisfies property (P2) as well, consider a pair of neighboring superclusters $(v_1, S_1, T_1), (v_2, S_2, T_2) \in \mathcal{C}'$, and an edge $(u_1, u_2) \in E$ between them such that $u_1 \in S_1$ and $u_2 \in S_2$. By step 2(a) of Procedure SC, $S_1, S_2 \subseteq \bigcup_{\tilde{S}_i \in \mathcal{C}} \tilde{S}_i$. Hence there exist clusters $\tilde{S}_i, \tilde{S}_j \in \mathcal{C}$ such that (a) $\tilde{S}_i \subseteq S_1$, (b) $\tilde{S}_j \subseteq S_2$, (c) $u_1 \in \tilde{S}_i$, and (d) $u_2 \in \tilde{S}_j$. Since $\mathcal{C}$ is adjacency-preserving with respect to H, it follows from (c) and (d) that there exists either a node $u_j \in \tilde{S}_j$ such that the edge (u_1, u_j) is in $E(H)$, or a node $u_i \in \tilde{S}_i$ such that the edge (u_i, u_2) is in $E(H)$. In either case, we are done by (a) and (b). □

2.3. The main algorithm. We proceed with the description of our main algorithm, named Algorithm SP_CONS. Our construction uses parameters κ, J, and Υ, to be fixed explicitly later on. The following analysis is valid for any nonnegative integers κ, J, and Υ that satisfy $1 \leq J \leq \lceil \log \kappa \rceil$ and $\Upsilon = \Omega((\kappa/2^{J-3})^J)$. Set $t_j = (\kappa - 2^{j-1}) \,/\, (2^{j-1}\kappa)$ and $\delta_j = \delta_j(\Upsilon, J) = \Upsilon^{j/J}$ for every $1 \leq j \leq J$. Also for every $1 \leq j \leq J-1$ set $\sigma_j = n^{t_{J-j} - t_{J-j+1}} = n^{2^{j-J}}$. Denote $\tau_j = [\kappa t_{J+1-j}, \kappa t_{J-j})$ for

Input: Graph $G=(V,E)$, integers $\kappa, J, \Upsilon > 1$.
Output: Subgraph H

1. $(\mathcal{G}, H) \leftarrow$ DOWN_PART(G);
2. $\mathcal{C}' \leftarrow \emptyset$;
3. **For** $j = 1$ to $J-1$ **do:**
   (a) $\mathcal{C} \leftarrow \mathcal{C}' \cup \bigcup_{i\in\tau_j} \mathcal{A}_i(\mathcal{G})$;
   (b) $(\mathcal{C}', H', \mathcal{R}) \leftarrow \mathrm{SC}(G, \mathcal{C}, \sigma_j, 2\delta_j(\Upsilon, J))$;
   (c) $E(H) \leftarrow E(H) \cup E(H')$;
4. $\mathcal{R} \leftarrow \mathcal{C}' \cup \bigcup_{i\in\tau_J} \mathcal{A}_i(\mathcal{G})$;
5. **For** every pair of spanned clusters (v_i, S_i, T_i), $(v_j, S_j, T_j) \in \mathcal{R}$ such that $dist_G(S_i, S_j) \le 2\delta_J(\Upsilon, J)$ **do:**
   (a) Calculate the shortest path P_{ij} between S_i and S_j;
   (b) $E(H) \leftarrow E(H) \cup E(P_{ij})$.
6. Return H as the resulting spanner.

FIG. 3. *Algorithm* SP_CONS.

$1 \le j \le J-1$ and $\tau_J = [\kappa t_1, \kappa)$. Observe that for $j \le J \le \lceil \log\kappa \rceil$, t_j is nonnegative. The parameters Υ and J will be chosen in a such a way that $\Upsilon^{1/J}$ will be a rather large constant, so the parameters δ_j for $1 \le j \le J$ are just the consecutive powers of this constant. These powers will serve as distance thresholds, and will grow as the algorithm proceeds.

As already mentioned, the algorithm starts by forming the ground partition $\mathcal{G}$. The ground partition contains a subset $\mathcal{Z} = \bigcup_{i<t_J\kappa} \mathcal{A}_i(\mathcal{G})$ of *singleton* clusters, that is, clusters that contain single vertex, and therefore have radius 0 (these properties of the clusters of the set $\mathcal{Z}$ will be ensured by the appropriate choice of the parameters J and κ). After forming the ground partition $\mathcal{G}$, the algorithm invokes iteratively Procedure SC. On each iteration the procedure forms a new spanned partition with fewer clusters (each of which is larger), and also takes care of clusters that were not merged into superclusters, by interconnecting pairs of nearby clusters by paths which are added to the subgraph H. More specifically, the input spanned partition $\mathcal{C}$ of Procedure SC on iteration j, $j = 1, 2, \ldots, J-1$, is the union of the output spanned partition $\mathcal{C}'$ of the previous $(j-1)$st iteration (for $j = 0$, $\mathcal{C}' = \emptyset$), and of the appropriate subset $Q_j = \bigcup_{i\in\tau_j} \mathcal{A}_i(\mathcal{G})$. The radii and the sizes of the clusters of Q_j grow with j. Algorithm SP_CONS repeats these iterations until we are left with a sufficiently small number of large clusters. The pairs of nearby clusters among these large clusters in the final spanned partition are again interconnected by shortest paths. These shortest paths are inserted into the subgraph H. At the end of this process, H contains the spanning trees of the shells of all the clusters and superclusters created through the process, and also all the shortest paths between clusters and superclusters that were created throughout the execution. Note that the notion of a "nearby" pair of clusters changes from one iteration of the algorithm to another. Specifically, on iteration j, pairs of clusters at distance $\delta_j(\Upsilon, \kappa)$ are considered close. This change corresponds to the increase of clusters radii.

A formal description of the algorithm is given in Figure 3. Figure 4 provides a schematic illustration.

Remark. The spanned partition $\mathcal{R}$ returned by Procedure SC is not used by the main algorithm; it is output by the procedure for the convenience of the analysis only.

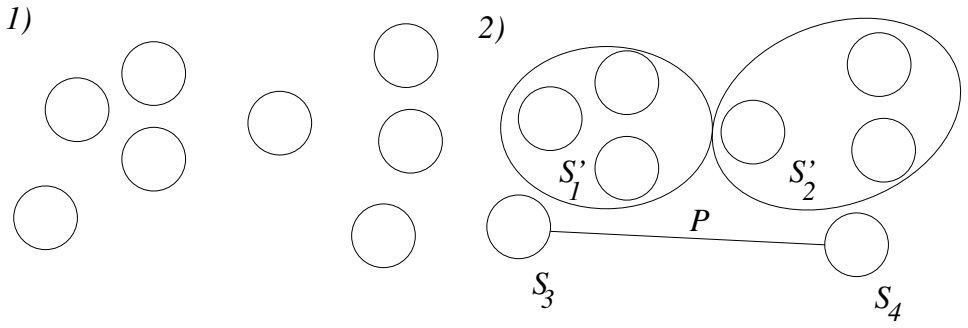

FIG. 4. 1) *The ground partition $\mathcal{G}$ is obtained by applying Procedure* DOWN_PART *to the graph G.* 2) *The two superclusters S'_1 and S'_2 were formed by an application of Procedure* SC. *The two clusters S_3 and S_4 were not merged into a supercluster. Thus, the shortest path P between them was inserted into the subgraph H.*

3. Analysis. To begin with, it is not hard to see that Algorithm SP_CONS runs in polynomial time. Indeed, each invocation of Procedure SC requires essentially at most n BFS explorations of the graph, and the same is true regarding Procedure DOWN_PART. The precise analysis of its running time is deferred to section 4.

We proceed with an analysis of the properties of the algorithm.

3.1. Bounding the cluster diameters. We first bound the diameters of the clusters and superclusters that are created in different iterations of Algorithm SP_CONS. For a spanned partition $\mathcal{C}$ denote $\hat{D}(\mathcal{C}) = \max_{(v,S,T)\in\mathcal{C}}\{diam(v,S,T)\}$ (for an empty spanned partition $\mathcal{C} = \emptyset$, let $\hat{D}(\mathcal{C}) = 0$). Since the graph G remains the same in all the invocations of Procedure SC, we henceforth omit it from its list of parameters.

LEMMA 3.1. *Let $\mathcal{C}'$ be spanned partition output by the invocation* $\mathrm{SC}(\mathcal{C},\sigma,\delta)$ *of Procedure* SC. *Then $\hat{D}(\mathcal{C}') \le 3\hat{D}(\mathcal{C}) + 2\delta$.*

Proof. Consider some supercluster $(v', S', T') \in \mathcal{C}'$. It was created at step 2(b)ii of Procedure SC. Therefore, it has the form of a star, with a cluster $(v' = v_0, S_0, T_0) \in \mathcal{C}$ in the middle, connected to some $b \ge \sigma$ clusters $(v_1, S_1, T_1), \ldots, (v_b, S_b, T_b) \in \mathcal{C}$ by shortest paths $P_1, \ldots, P_b$ of length at most δ. Consider a pair of vertices $z_i \in S_i$, $z_j \in S_j$ such that $1 \le i < j \le b$. Let $u_i \in S_i$ and $w_i \in S_0$ be the endpoints of the path P_i, and let $u_j \in S_j$ and $w_j \in S_0$ be the endpoints of the path P_j. Then

$$\begin{aligned} dist_{G(T')}(z_i,z_j) \le\ & dist_{G(T')}(z_i,u_i) + dist_{G(T')}(u_i,w_i) + dist_{G(T')}(w_i,w_j) \\ & + dist_{G(T')}(w_j,u_j) + dist_{G(T')}(u_j,z_j)\ . \end{aligned}$$

As by step 2(b)iiB of Procedure SC, $E(T_0), E(T_i), E(T_j) \subseteq E(T')$, and $z_i, u_i \in V(T_i)$, $w_i, w_j \in V(T_0)$ and $u_j, z_j \in V(T_j)$, it follows that

$$\begin{aligned} dist_{G(T')}(z_i,u_i) &\le dist_{G(T_i)}(z_i,u_i) \ \le\ diam(v_i,S_i,T_i)\ , \\ dist_{G(T')}(w_i,w_j) &\le dist_{G(T_0)}(w_i,w_j) \ \le\ diam(v_0,S_0,T_0)\ , \\ dist_{G(T')}(u_j,z_j) &\le dist_{G(T_j)}(u_j,z_j) \ \le\ diam(v_j,S_j,T_j)\ . \end{aligned}$$

Note that $diam(v_i,S_i,T_i), diam(v_0,S_0,T_0), diam(v_j,S_j,T_j) \le \hat{D}(\mathcal{C})$. Also, (by step 2(b)iiB of Procedure SC), $E(P_i), E(P_j) \subseteq E(T')$, and $u_i, w_i \in V(P_i)$, $u_j, w_j \in V(P_j)$,

implying that

$$dist_{G(T')}(u_i,w_i) \le dist_{G(P_i)}(u_i,w_i) \le \delta\ ,$$
$$dist_{G(T')}(u_j,w_j) \le dist_{G(P_j)}(u_j,w_j) \le \delta\ .$$

To conclude

$$\begin{aligned} dist_{G(T')}(z_i,z_j) &\le diam(v_i,S_i,T_i) + \delta + diam(v_0,S_0,T_0) + \delta + diam(v_j,S_j,T_j) \\ &\le 3\hat{D}(\mathcal{C}) + 2\delta. \end{aligned}$$

It is easy to see that this bound applies to the distance between any pair of vertices z_i, z_j in S'. □

For any $1 \le j \le J-1$, let $\mathcal{C}_j$ be the input partition of the jth invocation of Procedure SC and let $(\mathcal{C}'_j, H_j, \mathcal{R}_j)$ be the output returned by this invocation. The collection of clusters $\mathcal{R}_J$ is defined to be the set $\mathcal{R}$ created in step 4 of Algorithm Sp_Cons. We refer to clusters of $\mathcal{R}_j$ as the *jth level clusters* of the resulting partition. Note that these clusters are never merged again in subsequent iterations.

By steps 1 and 3 of Procedure SC, and since after step 1 of Procedure SC no cluster is inserted into the collection of uncovered spanners $\mathcal{U}$, we have that for every $1 \le j \le J-1$,

$$\mathcal{R}_j \ \subseteq\ \mathcal{C}_j. \tag{1}$$

Also, for every $1 \le j \le J-1$, since on the jth iteration of Algorithm Sp_Cons, Procedure SC is invoked with distance parameter δ_j, by Lemma 3.1

$$\hat{D}(\mathcal{C}'_j) \ \le\ 3\hat{D}(\mathcal{C}_j) + 2\delta_j. \tag{2}$$

Lemma 3.2. *For every integer $0 \le j \le J-2$,*

$$\mathcal{R}_{j+1} \subseteq \mathcal{C}_{j+1} \ =\ \mathcal{C}'_j \cup \bigcup_{i\in\tau_{j+1}} \mathcal{A}_i(\mathcal{G}).$$

Also, $\mathcal{R}_J \subseteq \mathcal{C}'_{J-1} \cup \bigcup_{i\in\tau_J} \mathcal{A}_i(\mathcal{G})$.

Proof. The first statement of the lemma follows by step 3(a) of Algorithm Sp_Cons and using (1). The second statement follows from the definition of $\mathcal{R}_J$ and step 4 of Algorithm Sp_Cons. □

Lemma 3.3. *For every integer $1 \le j \le J-1$,*

$$\text{(a)}\quad \hat{D}(\mathcal{C}'_j) \le 2\left(3^j\kappa t_{J-1} + \sum_{l=1}^{j} 3^{j-l}\Upsilon^{l/J}\right),$$
$$\text{(b)}\quad \hat{D}(\mathcal{C}_j) \le 2\left(3^{j-1}\kappa t_{J-1} + \sum_{l=1}^{j-1} 3^{j-1-l}\Upsilon^{l/J}\right).$$

Proof. By induction on j. For the induction base for $j=1$, note that the first invocation of Procedure SC is with $\mathcal{C} = \bigcup_{i\in\tau_1} \mathcal{A}_i(\mathcal{G})$. By definition of $\mathcal{A}_i(\mathcal{G})$ we have $\hat{D}(\mathcal{C}) < \hat{D}(\mathcal{A}_{t_{J-1}\kappa}(\mathcal{G})) \le 2t_{J-1}\kappa$, establishing claim (b). Claim (a) now follows by inequality (2).

For the induction step, assume the claims for j between 1 and $J-2$ and consider $j+1$. The parameter δ in the invocation of Procedure SC that formed the clusters of $\mathcal{C}'_{j+1}$ was equal to $\delta_{j+1} = \Upsilon^{(j+1)/J}$. By the induction hypothesis

$$\hat{D}(\mathcal{C}'_j) \;\leq\; 2\left(3^j \kappa t_{J-1} + \sum_{l=1}^{j} 3^{j-l}\Upsilon^{l/J}\right). \tag{3}$$

By step 3(a) of Algorithm Sp_Cons, $\mathcal{C}_{j+1} = \mathcal{C}'_j \cup \bigcup_{i\in\tau_{j+1}} \mathcal{A}_i(\mathcal{G})$. Hence

$$\hat{D}(\mathcal{C}_{j+1}) \;\leq\; \max\left\{\hat{D}(\mathcal{C}'_j), \hat{D}\left(\bigcup_{i\in\tau_{j+1}} \mathcal{A}_i(\mathcal{G})\right)\right\}. \tag{4}$$

By definition of $\mathcal{A}_i(\mathcal{G})$ and t_j

$$\hat{D}\Big(\bigcup_{i\in\tau_{j+1}} \mathcal{A}_i(\mathcal{G})\Big) \;\leq\; 2\kappa t_{J-(j+1)} \;\leq\; \frac{\kappa}{2^{J-j-3}}\,.$$

Recall that $\Upsilon^{1/J} = \Omega(\kappa/2^{J-3})$. Also, if Υ is set in such a way that $\Upsilon^{1/J}$ is a sufficiently large constant (e.g., $\Upsilon^{1/J} \geq 4$), then $2^j \leq \Upsilon^{(j-1)/J}$, and thus

$$\hat{D}\left(\bigcup_{i\in\tau_{j+1}} \mathcal{A}_i(\mathcal{G})\right) \;\leq\; \frac{\kappa}{2^{J-j-3}} \;\leq\; \Upsilon^{j/J} \;\leq\; \sum_{l=1}^{j} 3^{j-l}\Upsilon^{l/J}\,. \tag{5}$$

Now using inequalities (3), (4), and (5), we get

$$\hat{D}(\mathcal{C}_{j+1}) \;\leq\; 2\left(3^j \kappa t_{J-1} + \sum_{l=1}^{j} 3^{j-l}\Upsilon^{l/J}\right),$$

yielding claim (b). Also by inequality (2),

$$\begin{aligned}\hat{D}(\mathcal{C}'_{j+1}) \;&\leq\; 3\cdot 2\left(3^j \kappa t_{J-1} + \sum_{l=1}^{j} 3^{j-l}\Upsilon^{l/J}\right) + 2\Upsilon^{(j+1)/J}\\ &=\; 2\left(3^{j+1} \kappa t_{J-1} + \sum_{l=1}^{j+1} 3^{j+1-l}\Upsilon^{l/J}\right),\end{aligned}$$

yielding claim (a). ∎

Corollary 3.4. *For every integer $1 \leq j \leq J$,*

$$\hat{D}(\mathcal{R}_j) \;\leq\; 2\left(3^{j-1} \kappa t_{J-1} + \sum_{l=1}^{j-1} 3^{j-1-l}\Upsilon^{l/J}\right).$$

Proof. For $1 \leq j \leq J-1$ the claim follows from Lemma 3.3 by (1). For $j = J$, we get by step 4 of Algorithm Sp_Cons that

$$\hat{D}(\mathcal{R}_J) \;\leq\; \max\{\kappa, \hat{D}(\mathcal{C}'_{J-1})\} \;\leq\; 2\left(3^{J-1} \kappa t_{J-1} + \sum_{l=1}^{J-1} 3^{J-l}\Upsilon^{l/J}\right). \tag{6}$$

∎

3.2. Bounding the construction size. In this section we bound the size of the subgraph returned by Algorithm Sp_Cons. First, by Lemma 2.2, the number of edges inserted into the subgraph H by Procedure Down_Part (i.e., by step 1 of Algorithm Sp_Cons) is $O(n^{1+1/\kappa})$. Next, we analyze the number of edges inserted into the subgraph H at step 3 of Algorithm Sp_Cons.

As each of the superclusters created by Procedure SC contains at least σ (disjoint) clusters of the input partition $\mathcal{C}$, we have the following.

Lemma 3.5. *Let $\mathcal{C}'$ denote the spanned partition output by an invocation* $\mathrm{SC}(\mathcal{C},\sigma,\delta)$ *of Procedure* SC. *Then $\check{S}(\mathcal{C}') \geq \sigma \cdot \check{S}(\mathcal{C})$.*

Lemma 3.6. *Let H denote the subgraph output by an invocation of Procedure* $\mathrm{SC}(\mathcal{C},\sigma,\delta)$. *Then $|E(H)| = O(n\sigma\delta/\check{S}(\mathcal{C}))$.*

Proof. Note that by definition of spanned partition, $\mathcal{C}$ is a collection of disjoint clusters. Hence $|\mathcal{C}| \leq n/\check{S}(\mathcal{C})$.

Consider the supergraph G' in which every vertex represents a cluster of $\mathcal{C}$ and there is an edge between two vertices if and only if the corresponding two clusters are at distance of at most δ in G. Step 2 of Procedure SC creates a forest $\mathcal{F}$ of disjoint star-like trees covering a subset of vertices of G'. It then inserts into H the spanning trees of all the clusters of $\mathcal{C}$ corresponding to the vertices of G' covered by $\mathcal{F}$ (summing up to $O(n)$ edges) and the paths in G corresponding to the edges of $\mathcal{F}$. As $\mathcal{F}$ is a forest,

$$|E(\mathcal{F})| \;\leq\; |V(\mathcal{F})| \;\leq\; |\mathcal{C}| \;\leq\; n/\check{S}(\mathcal{C})\;.$$

Also each path represented by an edge of $\mathcal{F}$ has length of at most δ. Hence step 2 of Procedure SC inserts at most $n\delta/\check{S}(\mathcal{C})$ edges into H.

Note that after removing from G' the vertices covered by $\mathcal{F}$, the removed supergraph has maximal degree of at most σ. Hence the number of remaining edges is at most $|\mathcal{C}|\sigma \leq n\sigma/\check{S}(\mathcal{C})$. Step 4 inserts into H all the paths corresponding to the edges left in G'. Hence it inserts at most $n\sigma\delta/\check{S}(\mathcal{C})$ edges.

It follows that overall, the number of edges inserted by Procedure SC is at most

$$n\sigma\delta/\check{S}(\mathcal{C}) + n\delta/\check{S}(\mathcal{C}) + n \;=\; O(n\sigma\delta/\check{S}(\mathcal{C})). \qquad \square$$

Lemma 3.7. *For any $1 \leq j \leq J-1$, $\check{S}(\mathcal{C}_j) \geq n^{t_{J-j+1}}$.*

Proof. The proof is by induction on j. The induction base is $j=1$. Recall that $\mathcal{G}$ is the ground partition returned by Procedure Down_Part. In iteration 1,

$$\check{S}(\mathcal{C}_1) \;=\; \check{S}\left(\bigcup_{i\in\tau_1}\mathcal{A}_i(\mathcal{G})\right) \;=\; \check{S}(\mathcal{A}_{t_J\kappa}(\mathcal{G})) \geq n^{t_J}$$

by Lemma 2.1.

For the induction step, let $1 < j < J-1$, and assume (by the inductive hypothesis) that $\check{S}(\mathcal{C}_j) \geq n^{t_{J-j+1}}$. Since $\sigma_j = n^{t_{J-j}-t_{J+1-j}}$, Lemma 3.5 implies that

$$\check{S}(\mathcal{C}'_j) \;\geq\; \check{S}(\mathcal{C})\cdot\sigma_j \geq n^{t_{J+1-j}}\cdot n^{t_{J-j}-t_{J+1-j}} = n^{t_{J-j}}.$$

Also $\mathcal{C}_{j+1} = \mathcal{C}_j \cup \bigcup_{i\in\tau_{j+1}}\mathcal{A}_i(\mathcal{G})$. Hence $\check{S}(\mathcal{C}_{j+1}) \geq \min\{\check{S}(\mathcal{C}'_j), \check{S}(\bigcup_{i\in\tau_{j+1}}\mathcal{A}_i(\mathcal{G}))\}$. Since by Lemma 2.1, $\check{S}(\bigcup_{i\in\tau_{j+1}}\mathcal{A}_i(\mathcal{G})) \geq n^{t_{J-j}}$, the lemma follows. $\square$

Lemma 3.8. *At the end of step* 3 *of Algorithm* Sp_Cons, *$|E(H)| = O(\Upsilon n^{1+1/\kappa})$.*

Proof. By Lemma 3.7, $|\mathcal{C}_j| \leq n/\check{S}(\mathcal{C}_j) \leq n^{1-t_{J-j+1}}$. Hence by Lemma 3.6, in the jth iteration, Procedure SC inserts into the output subgraph H_j no more than $\delta_j n^{1-t_{J-j+1}}\sigma_j$ edges. Hence

$$|E(H_j)| \leq n^{1-t_{J-j+1}} n^{t_{J-j}-t_{J-j+1}}\delta_j \;=\; n^{1+t_{J-j}-2t_{J-j+1}}\delta_j \;. \tag{7}$$

Observe that the exponent is

$$1 + t_{J-j} - 2t_{J-j+1} \;=\; 1 + \frac{\kappa - (2^{J-j-1}-1)}{2^{J-j-1}\kappa} - 2\cdot\frac{\kappa-(2^{J-j}-1)}{2^{J-j}\kappa} \;=\; \frac{\kappa+1}{\kappa}\;.$$

Thus $|E(H_j)| \;=\; O(n^{1+1/\kappa}\Upsilon^{j/J})$. Hence the size of the subgraph H generated at the end of step 3 of Algorithm Sp_Cons can be bounded by

$$|E(H)| \;\leq\; \sum_{j=1}^{J}|E(H_j)| \;=\; O(n^{1+1/\kappa})\sum_{j=1}^{J}(\Upsilon^{1/J})^j \;=\; O(\Upsilon n^{1+1/\kappa})\;. \qquad \square$$

Lemma 3.9. *Step* 5 *of Algorithm* Sp_Cons *inserts at most* $O(\Upsilon n^{2/\kappa})$ *additional edges to the subgraph* H.

Proof. By Lemma 3.2,

$$\mathcal{R}_J \;\subseteq\; \mathcal{C}'_{J-1} \cup \bigcup_{i\in\tau_J}\mathcal{A}_i(\mathcal{G})\;,$$

and thus

$$\check{S}(\mathcal{R}_J) \;\geq\; \min\left\{\check{S}(\mathcal{C}'_{J-1}), \check{S}\left(\bigcup_{i\in\tau_J}\mathcal{A}_i(\mathcal{G})\right)\right\} \;=\; \min\left\{\check{S}(\mathcal{C}'_{J-1}), \min_{i\geq\kappa t_1}\check{S}(\mathcal{A}_i(\mathcal{G}))\right\}\;.$$

By Lemma 3.5, $\check{S}(\mathcal{C}'_{J-1}) \geq n^{t_2}n^{t_1-t_2} = n^{t_1}$. By Lemma 2.1, $\min_{i\geq\kappa t_1}\{\check{S}(\mathcal{A}_i(\mathcal{G}))\} \geq n^{t_1}$. Hence $\check{S}(\mathcal{R}_J) \geq n^{t_1}$. Thus the number of pairs of such clusters is no greater than $n^{2-2t_1} = n^{2/\kappa}$. Since the path inserted into the output subgraph between each pair is of length at most $2\cdot\Upsilon$, the total number of edges inserted is at most $O(\Upsilon n^{2/\kappa})$. $\square$

Combining Lemmas 3.8 and 3.9, we have the following.

Corollary 3.10. *The size of the subgraph output by Algorithm* Sp_Cons *is* $O(\Upsilon n^{1+1/\kappa})$.

3.3. Stretch analysis. In this section we bound the stretch of the spanner H output by Algorithm Sp_Cons. This is done by considering a pair of nodes $u, w \in V$, and one of the shortest paths P between u and w in G. This path is partitioned to segments of length no longer than Υ^J. It is convenient to visualize this path as going from left to right, from u to w (see Figure 5).

Consider some segment P' and let u' (resp., w') be its left (resp., right) endpoint. For a set X of clusters, a node u (resp., a path P) is said to be *X-clustered* if $u \in C$ (resp., $V(P) \subseteq C$) for some cluster C of X. Let u_J (resp., w_J) be the leftmost (resp., rightmost) $\mathcal{R}_J$-clustered node of P', and let C_{u_J} and C_{w_J} be the clusters such that $u_J \in C_{u_J}$ and $w_J \in C_{w_J}$. Since $dist_G(C_{u_J}, C_{w_J}) \leq dist_G(u_J, w_J) \leq \Upsilon$, there is a path of length $dist_G(C_{u_J}, C_{w_J})$ between some nodes $u'' \in C_{u_J}$ and $w'' \in C_{w_J}$ in the spanner H. It follows that there is a path of length no longer than $dist_G(u_J, w_J) + diam(C_{u_J}) + diam(C_{w_J})$ between u_J and w_J in H.

Next, we consider the subpaths from u to u_J, and from w_J from w for every segment of P, and observe that these subpaths are $(\mathcal{Z} \cup \bigcup_{i=1}^{J-1}\mathcal{R}_i)$-clustered. We partition these subpaths into subsegments of length $\Upsilon^{(J-1)/J}$, on each subsegment

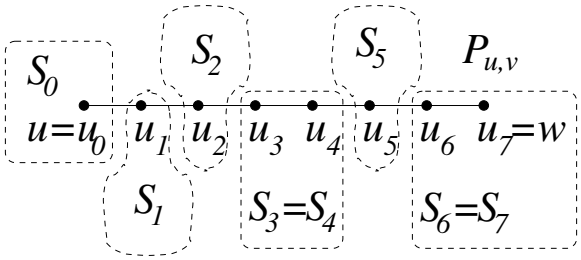

FIG. 5. *The path $P_{u,w}$ and the clusters along it.*

find the leftmost and the rightmost $\mathcal{R}_{J-1}$-clustered nodes and use the "short" paths between their clusters in the spanner. This argument is repeated recursively J times, until we are left with $\mathcal{Z}$-clustered subpaths, whose stretch can be easily bounded.

We next present a rigorous argument formalizing the above intuitive description. First, the following lemma can be proved using Lemmas 2.4 and 2.5 by a straightforward induction on the number of iterations of the algorithm.

LEMMA 3.11. *All the spanned partitions created throughout Algorithm* SP_CONS *are adjacency-preserving with respect to the resulting subgraph H returned by the algorithm.*

Next, we establish the following property of adjacency-preserving partitions that will be later used in the stretch analysis.

LEMMA 3.12. *Let $u, w \in V$ be a pair of nodes in the graph $G = (V, E)$, and let $P_{u,w} = (u = u_0, u_1, \ldots, u_x = w) \subseteq E$ be one of the shortest paths between them. Let H be a subgraph of G with $E(H) \subseteq E$, and let $\mathcal{S}$ be an adjacency-preserving spanned partition with respect to H, such that $V(P_{u,w}) \subseteq \bigcup_{(v,S,T)\in\mathcal{S}} S$. For $i = 0, 1, \ldots, x$, let $(v_i, S_i, T_i) \in \mathcal{S}$ be spanned clusters such that $u_i \in S_i$. (This is well-defined, since $V(P_{u,w}) \subseteq \bigcup_{(v,S,T)\in\mathcal{S}} S$, and the clusters of $\mathcal{S}$ are disjoint, by definition of spanned partition.) Finally, let $u'_0 \in S_0$, $u'_x \in S_x$ be some nodes. Then*

$$dist_H(u'_0, u'_x) \;\leq\; \sum_{i=0}^{x} (diam(v_i, S_i, T_i) + 1) - 1 \;.$$

Proof. The proof is by induction on x. The induction base is $x = 0$. Then $\{S_0, \ldots, S_x\} = \{S_0\}$, and $u = w \in S_0$. Let $u'_0, u'_x \in S_0$ be some arbitrary nodes in this cluster. Since $T_0 \subseteq H$, $dist_H(u'_0, u'_x) \leq diam(v_0, S_0, T_0)$, as required.

For the induction step, assume that the statement of the lemma is true for some $x \geq 0$. Consider the pair of neighboring clusters S_x, S_{x+1}. The edge (u_x, u_{x+1}) between them is spanned either through S_x or through S_{x+1}. In the former case, there exists a node $u'_x \in S_x$ such that the edge (u'_x, u_{x+1}) is in H. In the latter case, there exists a node u'_{x+1} such that the edge (u_x, u'_{x+1}) is in H.

Consider some pair of nodes $u''_0 \in S_0$, $u''_{x+1} \in S_{x+1}$. In the case when the edge (u_x, u_{x+1}) is spanned through S_x, using the induction hypothesis for the pair of nodes $u''_0 \in S_0$ and $u'_x \in S_x$, it follows that

$$\begin{aligned} dist_H(u''_0, u''_{x+1}) &\leq dist_H(u''_0, u'_x) + dist_H(u'_x, u_{x+1}) + dist_H(u_{x+1}, u''_{x+1}) \\ &\leq \left(\sum_{i=0}^{x} (diam(v_i, S_i, T_i) + 1) - 1 \right) + 1 + diam(v_{x+1}, S_{x+1}, T_{x+1}) \\ &= \sum_{i=0}^{x+1} (diam(v_i, S_i, T_i) + 1) - 1 \;. \end{aligned}$$

The case when the edge (u_x, u_{x+1}) is spanned through S_{x+1} follows symmetrically. □

We use the following notions. For $0 \leq j \leq J$, a path P in G is called a *class-j path* if it contains only $(\mathcal{Z} \cup \bigcup_{i=1}^{j} \mathcal{R}_i)$-clustered vertices (in particular, P is a class-0 path if all its vertices are $\mathcal{Z}$-clustered). A pair of vertices u, w is *j-reachable* if there is a shortest path between them that is a class-j path.

For every $1 \leq j \leq J$ denote

$$\gamma_j \;=\; \sum_{i=1}^{j} 2^i \cdot \hat{D}(\mathcal{R}_{j-i+1}).$$

For a real r, let $down(r) = \lceil r \rceil - 1$.

LEMMA 3.13. *For every integer $1 \leq j \leq J$, and for every j-reachable pair of vertices u'_j, u''_j,*

$$dist_H(u'_j, u''_j) \;\leq\; dist_G(u'_j, u''_j) \left(2 \cdot down(t_J \kappa) + 1 + \sum_{i=1}^{j} \frac{\gamma_i}{\Upsilon^{i/J}} \right) + \gamma_j \ .$$

Proof. Let P be a class-j path in G between u'_j and u''_j. It is convenient to visualize the path P as going from left to right, with the vertex u'_j at the leftmost end and the vertex u''_j the rightmost end. We prove by induction on j a claim that is slightly stronger than the statement of the lemma. Specifically, let S'_j, S''_j be the clusters such that $u'_j \in S'_j$, $u''_j \in S''_j$. Let $v'_j \in S'_j$, $v''_j \in S''_j$ be arbitrary nodes. Then

$$dist_H(v'_j, v''_j) \;\leq\; dist_G(u'_j, u''_j) \left(2 \cdot down(t_J \kappa) + 1 + \sum_{i=1}^{j} \frac{\gamma_i}{\Upsilon^{i/J}} \right) + \gamma_j \ .$$

Induction base ($j = 1$): We claim that

$$dist_H(v'_1, v''_1) \;\leq\; dist_G(u'_1, u''_1) \left(2 \cdot down(t_J \kappa) + 1 + \frac{\gamma_1}{\Upsilon^{1/J}} \right) + \gamma_1 \tag{8}$$

for a pair of 1-reachable vertices $u'_1 \in S'_1, u''_1 \in S''_1$ and any pair of nodes $v'_1 \in S'_1$, $v''_1 \in S''_1$.

We separate the discussion to two cases.

Case 1. $dist_G(u'_1, u''_1) \leq 2\Upsilon^{1/J}$.

Case 1.1. If the path P is a class-0 path, then it contains only $\mathcal{Z}$-clustered vertices. Note that $\mathcal{Z} = \bigcup_{i < t_J \kappa} \mathcal{A}_i(\mathcal{G}) = \bigcup_{i=0}^{down(t_J \kappa)} \mathcal{A}_i(\mathcal{G})$. Denote $z = down(t_J \kappa)$. By Lemmas 3.11 and 3.12,

$$\begin{aligned} dist_H(v'_1, v''_1) &\leq (dist_G(u'_1, u''_1) + 1) \left(\hat{D}(\mathcal{Z}) + 1 \right) - 1 \\ &\leq dist_G(u'_1, u''_1)(2z + 1) + 2z \ . \end{aligned}$$

Case 1.2. If the path P contains only one $\mathcal{R}_1$-clustered vertex, then by Lemmas 3.11 and 3.12,

$$dist_H(v'_1, v''_1) \;\leq\; dist_G(u'_1, u''_1)(2z + 1) + \hat{D}(\mathcal{R}_1) \ ,$$

and, again, we are done.

Case 1.3. It therefore remains to consider only the case where at least two vertices in the path P are $\mathcal{R}_1$-clustered. Let l_2 be the distance between the leftmost $\mathcal{R}_1$-clustered vertex, $w'_1 \in S'_1$, and the rightmost $\mathcal{R}_1$-clustered vertex, $w''_1 \in S''_1$, of P. Hence the two subpaths, from v'_1 to w'_1 and from w''_1 to v''_1, are spanned with multiplicative stretch of at most $(\hat{D}(\mathcal{Z})+1)$. The distance between S'_1 and S''_1 in G is at most $2\Upsilon^{1/J} = 2\delta_1(\Upsilon, J)$. Since at iteration 1 of Algorithm Sp_Cons, at step 4 of Procedure SC the shortest paths between all the pairs of clusters from $\mathcal{R}_1$ that are at distance at most $2\delta_1(\Upsilon, J)$ one from another were inserted into the subgraph H, it follows that there exist nodes $z'_1 \in S'_1$, $z''_1 \in S''_1$ such that $dist_H(z'_1, z''_1) = dist_G(w'_1, w''_1) = l_2$. By Lemmas 3.11 and 3.12,

$$dist_H(v'_1, z'_1) \le dist_G(u'_1, w'_1)(2z+1) + \hat{D}(\mathcal{R}_1) \ ,$$
$$dist_H(z''_1, v''_1) \le dist_G(w''_1, u''_1)(2z+1) + \hat{D}(\mathcal{R}_1) \ .$$

Hence

$$\begin{aligned} dist_H(v'_1, v''_1) &\le (dist_G(u'_1, u''_1) - l_2)(2z+1) + 2\hat{D}(\mathcal{R}_1) + l_2 \\ &\le dist_G(u'_1, u''_1)(2z+1) + 2\hat{D}(\mathcal{R}_1) \ . \end{aligned} \tag{9}$$

Case 2. $dist_G(u'_1, u''_1) > 2\Upsilon^{1/J}$. By partitioning the path P into segments of length $\Upsilon^{1/J}$, we get $dist_G(u'_1, u''_1) = (a-1)\Upsilon^{1/J} + \Upsilon'$, where $\Upsilon^{1/J} < \Upsilon' < 2\Upsilon^{1/J}$ and $a = \lfloor dist_G(u'_1, u''_1)/\Upsilon^{1/J} \rfloor$. Applying the previous argument to each segment separately, it follows that

$$\begin{aligned} dist_H(v'_1, v''_1) &\le (a-1)\left(\Upsilon^{1/J}(2z+1) + 2\hat{D}(\mathcal{R}_1)\right) + \Upsilon'(2z+1) + 2\hat{D}(\mathcal{R}_1) \\ &= dist_G(u'_1, u''_1)(2z+1) + 2a \cdot \hat{D}(\mathcal{R}_1) \\ &\le dist_G(u'_1, u''_1)(2z+1) + \frac{dist_G(u'_1, u''_1) \cdot 2\hat{D}(\mathcal{R}_1)}{\Upsilon^{1/J}} \\ &= dist_G(u'_1, u''_1)\left(2z+1+\frac{\gamma_1}{\Upsilon^{1/J}}\right) \ . \end{aligned} \tag{10}$$

As $\gamma_1 = 2\hat{D}(\mathcal{R}_1)$, the expression (8) dominates both (9) and (10), completing the proof of the induction base.

Induction step: Assume the induction hypothesis for some integer $1 < j \le J-1$. Consider a class-$(j+1)$ path P connecting u' and u'' in G. Let v' (resp., v'') be an arbitrary node in the same cluster as u' (resp., u'').

Case 1. P is of length no greater than $\Upsilon^{(j+1)/J}$.

We break the discussion into three subcases.

Case 1.1. If no $\mathcal{R}_{j+1}$-clustered vertex appears in the path P, we apply the induction hypothesis and we are done.

Case 1.2. Exactly one $\mathcal{R}_{j+1}$-clustered vertex w appears on P. Let w' be the left-hand neighbor of w and let w'' be the right-hand neighbor of w. Denote by S (resp., S'; S'') the cluster that contains w (resp., w'; w''). Let P' (resp., P'') be the path between u' and w' (resp., w'' and u''). Note that both subpaths P' and P'' are class-j paths (see Figure 6). Hence, the induction hypothesis is applicable to these subpaths.

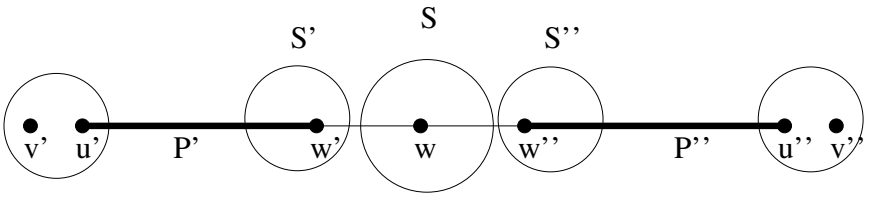

FIG. 6. *The solid lines represent class-j paths P' and P''. The cluster S is in $\mathcal{R}_{j+1}$.*

Therefore,

$$\begin{aligned} dist_H(v',v'') &\le dist_H(v',w') + dist_H(w',w'') + dist_H(w'',v'') \\ &\le (dist_G(u',u'') - 2)\left(2z + 1 + \sum_{i=1}^{j} \frac{\gamma_i}{\Upsilon^{i/J}}\right) \\ &\quad +2\gamma_j + dist_H(w',w'') \ . \end{aligned}$$

The analysis decomposes again to three subsubcases.

Case 1.2.1. The first is that both edges (w',w) and (w,w'') are spanned through the cluster S, i.e., there exist nodes $z,y \in S$ such that the edges (w',z) and (y,w'') are in H. Then

$$\begin{aligned} dist_H(w',w'') &\le dist_H(w',z) + dist_H(z,y) + dist_H(y,w'') \\ &\le 2 + \hat{D}(\mathcal{R}_{j+1}) \ . \end{aligned}$$

Case 1.2.2. The edge (w',w) is spanned through the cluster S', and the edge (w,w'') is spanned through the cluster S''. In this situation there exist nodes $z' \in S'$, $z'' \in S''$ such that the edges (z',w) and (w,z'') are in H. Thus $dist_H(w',w'') \le dist_H(w',z') + dist_H(z',z'') + dist_H(z'',w'')$. However, we observe that the induction hypothesis is applicable to the pairs of nodes v',z' and z'',v'' as well. Note also that $dist_H(z',z'') = 2$. Hence

$$\begin{aligned} dist_H(v',v'') &\le dist_H(v',z') + dist_H(z',z'') + dist_H(z'',v'') \\ &\le (dist_G(u',u'') - 2)\left(2z + 1 + \sum_{i=1}^{j} \frac{\gamma_i}{\Upsilon^{i/J}}\right) + 2\gamma_j + 2 \ . \end{aligned}$$

Case 1.2.3. The edge (w',w) is spanned through the cluster S', and the edge (w,w'') is spanned through the cluster S (the situation when the edge (w',w) is spanned through the cluster S, and the edge (w,w'') is spanned through the cluster S'' is symmetrical to this one). In this subcase there exist nodes $z' \in S'$, $z \in S$ such that the edges (z',w) and (z,w'') are in H. Hence we may apply the induction hypothesis to the pairs v',z' and w'',v'', and get

$$\begin{aligned} dist_H(v',v'') &\le (dist_G(u',u'') - 2)\left(2z + 1 + \sum_{i=1}^{j} \frac{\gamma_i}{\Upsilon^{i/J}}\right) \\ &\quad +2\gamma_j + 2 + \hat{D}(\mathcal{R}_{j+1}) \ . \end{aligned}$$

In all these three subsubcases,

$$dist_H(v',v'') \;\le\; dist_G(u',u'')\left(2z+1+\sum_{i=1}^{j}\frac{\gamma_i}{\Upsilon^{i/J}}\right)+2\gamma_j+\hat{D}(\mathcal{R}_{j+1})\ .$$

This expression is smaller than the bound in Lemma 3.13 with $j+1$ substituted for j, and so we are done in Case 1.2 too.

Case 1.3. There are at least two $\mathcal{R}_{j+1}$-clustered vertices in P. Let w (resp., z) be the leftmost (resp., rightmost) such vertex. Again, let w' (resp., z'') denote the left-hand (resp., right-hand) neighbor of w (resp., z). Denote by S_w, S_z, S', and S'' the clusters such that $w\in S_w$, $z\in S_z$, $w'\in S'$, and $z''\in S''$. Observe that H contains a path of length $dist_G(S_w,S_z)\le dist_G(w,z)$ between S_w and S_z. Similarly to the analysis of Case 1.2, where there was only one $\mathcal{R}_j$-clustered node in P, we decompose the analysis to subcases depending on whether the edge (w',w) is spanned through S_w or S', and on whether the edge (z,z'') is spanned through S_z or S''. Like in that situation, we apply the induction hypothesis on the subpaths between v' and w' (or some other appropriate node in S') and between z'' (or some other appropriate node in S'') and v''. Recall that $dist_G(S_w,S_z)\le 2\Upsilon^{(j+1)/J}=2\delta_{j+1}(\Upsilon,J)$. Also, at $(j+1)$st iteration of Algorithm Sp_Cons, at step 4 of Procedure SC (if $j<J-1$; if $j=J-1$ then at step 4 of Algorithm Sp_Cons) one of the shortest paths between S_w and S_z in G was inserted into H. Thus

$$\begin{aligned} dist_H(v',v'') &\le (dist_G(u',u'')-dist_G(w,z))\left(2z+1+\sum_{i=1}^{j}\frac{\gamma_i}{\Upsilon^{i/J}}\right)\\ &\quad+2\gamma_j+2\hat{D}(\mathcal{R}_{j+1})+dist_G(w,z)\\ &\le dist_G(u',u'')\left(2z+1+\sum_{i=1}^{j}\frac{\gamma_i}{\Upsilon^{i/J}}\right)+\gamma_{j+1}\ . \end{aligned}\tag{11}$$

Case 2. $dist_G(u',u'')>2\Upsilon^{(j+1)/J}$. In this situation, we partition the path into segments of length $\Upsilon^{(j+1)/J}$ each, except the last segment which may be of length between $\Upsilon^{(j+1)/J}$ and $2\Upsilon^{(j+1)/J}$. We next show that in this situation γ_{j+1} divided by $\Upsilon^{(j+1)/J}$ is introduced into the multiplicative term. Formally, $dist_G(u',u'')=(a-1)\Upsilon^{(j+1)/J}+\Upsilon'$, where $\Upsilon^{(j+1)/J}<\Upsilon'<2\Upsilon^{(j+1)/J}$ and $a=\lfloor dist_G(u_1',u_1'')/\Upsilon^{1/J}\rfloor$. Then

$$\begin{aligned} dist_H(v',v'') &\le (a-1)\left(\left(2z+1+\sum_{i=1}^{j}\frac{\gamma_i}{\Upsilon^{i/J}}\right)\Upsilon^{(j+1)/J}+\gamma_{j+1}\right)\\ &\quad+\left(2z+1+\sum_{i=1}^{j}\frac{\gamma_i}{\Upsilon^{i/J}}\right)\Upsilon'+\gamma_{j+1}\\ &=\left(2z+1+\sum_{i=1}^{j}\frac{\gamma_i}{\Upsilon^{i/J}}\right)\left((a-1)\Upsilon^{(j+1)/J}+\Upsilon'\right)+\gamma_{j+1}a\\ &\le dist_G(u',u'')\left(2z+1+\sum_{i=1}^{j}\frac{\gamma_i}{\Upsilon^{i/J}}\right)+\gamma_{j+1}\ \frac{dist_G(u',u'')}{\Upsilon^{(j+1)/J}}\\ &=dist_G(u',u'')\left(2z+1+\sum_{i=1}^{j+1}\frac{\gamma_i}{\Upsilon^{i/J}}\right)\ . \end{aligned}\tag{12}$$

In summary, for any $(j+1)$-reachable pair of vertices $u' \in S'$, $u'' \in S''$, and for any pair of nodes $v' \in S'$, $v'' \in S''$ the bound $dist_H(v', v'') \leq dist_G(u', u'')(2z + 1 + \sum_{i=1}^{j+1} \frac{\gamma_i}{\Upsilon^{i/J}}) + \gamma_{j+1}$ dominates both former bounds (11) and (12) on $dist_H(v', v'')$. ∎

Observe that any path in G is a class-J path. Hence by Lemma 3.13 we have the following corollary.

COROLLARY 3.14. *For any pair of vertices u', u'', $dist_H(u', u'') \leq \alpha \cdot dist_G(u', u'') + \beta$ for $\alpha = 2 \cdot down(t_J\kappa) + 1 + \sum_{i=1}^{J} \frac{\gamma_i}{\Upsilon^{i/J}}$ and $\beta = \gamma_J$.*

LEMMA 3.15. *For Υ and J such that $\Upsilon^{1/J} \geq 6$ and for any $1 \leq j \leq J$, $\gamma_j = O(t_{J-1}\kappa \cdot 3^j + \Upsilon^{(j-1)/J})$.*

Proof. By the definition of γ_j, $\gamma_j = \sum_{i=0}^{j-1} 2^{j-i} \hat{D}(\mathcal{R}_{i+1})$ for any $1 \leq j \leq J$. Next substitute the explicit formula for $\hat{D}(\mathcal{R}_j)$ given in Corollary 3.4 and get (dividing the expression by 4 for convenience)

$$\frac{\gamma_j}{4} \leq 2^{j-1} t_{J-1}\kappa + 2^{j-2}\left(3t_{J-1}\kappa + \Upsilon^{1/J}\right) + \cdots + 2^0\left(3^{j-1}\kappa t_{J-1} + \sum_{i=1}^{j-1} 3^{j-1-i}\Upsilon^{i/J}\right)$$

$$= t_{J-1}\kappa \sum_{i=0}^{j-1} 3^i 2^{j-1-i} + \sum_{l=1}^{j-1}\left(\Upsilon^{l/J} \sum_{i=0}^{j-l-1} 3^i 2^{j-l-1-i}\right). \tag{13}$$

Note that for any integer $p \geq 1$, $\sum_{i=0}^{p} 3^i 2^{p-i} < 3^p \frac{1}{1-2/3} = 3^{p+1}$. Since $\Upsilon^{1/J} \geq 6$,

$$\begin{aligned} \gamma_j/4 &\leq t_{J-1}\kappa 3^j + \sum_{i=1}^{j-1} \Upsilon^{i/J} 3^{j-i} \leq t_{J-1}\kappa 3^j + \Upsilon^{(j-1)/J} 3 \sum_{i=0}^{j-2}\left(\frac{3}{\Upsilon^{1/J}}\right)^i \\ &< t_{J-1}\kappa 3^j + 6\Upsilon^{(j-1)/J}. \end{aligned}$$ ∎

LEMMA 3.16. *For Υ and J such that $\Upsilon^{1/J} \geq 6$,*

1. *the additive term is $\beta = O(t_{J-1}\kappa \cdot 3^J + \Upsilon^{(J-1)/J})$,*
2. *the multiplicative factor is bounded by $\alpha = 1 + 2 \cdot down(t_J\kappa) + O(\frac{J + t_{J-1}\kappa}{\Upsilon^{1/J}})$.*

Proof. By Corollary 3.14 and substituting $j = J$ in Lemma 3.15, we get

$$\begin{aligned} \alpha &\leq 1 + 2 \cdot down(t_J\kappa) + \sum_{j=1}^{J} \frac{\gamma_j}{\Upsilon^{j/J}} \\ &\leq 1 + 2 \cdot down(t_J\kappa) + O\left(\frac{J}{\Upsilon^{1/J}}\right) + O\left(t_{J-1}\kappa \sum_{j=1}^{J} \frac{3^j}{\Upsilon^{j/J}}\right) \\ &\leq 1 + 2 \cdot down(t_J\kappa) + O\left(\frac{J}{\Upsilon^{1/J}}\right) + O\left(t_{J-1}\kappa \cdot \frac{1}{\Upsilon^{1/J}} \cdot \frac{1}{1 - 3/\Upsilon^{1/J}}\right) \\ &= 1 + 2 \cdot down(t_J\kappa) + O\left(\frac{J + t_{J-1}\kappa}{\Upsilon^{1/J}}\right) \end{aligned}$$

and $\beta = O(t_{J-1}\kappa 3^J + \Upsilon^{(J-1)/J})$. This completes the proof of Lemma 3.16. ∎

THEOREM 3.17. *For any fixed $0 < \epsilon < 1$ and fixed integer $2 \leq \kappa = O(\log n)$ there exists a fixed $\beta = \beta(\kappa, \epsilon) = \kappa^{\max\{\log\log\kappa - \log\epsilon, 3\}}$ such that for any graph G, running Algorithm* SP_CONS *on G, κ, $J = \log\kappa$ and β yields a $(1+\epsilon, \beta)$-spanner of G with $O(\beta n^{1+1/\kappa})$ edges.*

Proof. Since $t_i = (\kappa - 2^{i-1}) \,/\, (2^{i-1}\kappa)$, it is sufficient to set $J = \lceil \log \kappa \rceil$ in order to ensure $t_J \kappa \leq 1$, i.e., $down(t_J \kappa) = 0$. Therefore, substituting $J = \lceil \log \kappa \rceil$ into Lemma 3.16 implies that the multiplicative factor of the stretch is $\alpha = 1 + O(\frac{\log \kappa}{\Upsilon^{1/\log \kappa}})$ and the additive term is $\beta = O(\Upsilon^{(\log \kappa - 1)/\log \kappa} + \kappa^{\log 3})$. To get a multiplicative stretch of $1+\epsilon$ for any fixed $0 < \epsilon < 1$ and to ensure that $\Upsilon^{1/J} \geq \kappa/2^{J-3} = 8$, we set $\Upsilon^{1/\log \kappa} = \max\{\epsilon^{-1} \log \kappa, 8\}$, or $\Upsilon = \kappa^{\max\{\log\log \kappa - \log \epsilon, 3\}}$. The theorem now follows by Corollary 3.10. ∎

Note that for $\kappa = \Theta(\log n)$ the size of the spanner becomes $O(\beta(\kappa,\epsilon)n) = o(n^{1+\nu})$ for any $\nu > 0$, and so there is no reason to consider values of κ greater than $O(\log n)$.

COROLLARY 3.18. *For any constant $\epsilon > 0$, $\lambda > 0$ there exists a constant $\beta'(\epsilon,\lambda)$ such that for any n-vertex graph $G = (V,E)$ there exists a polynomial time constructible subgraph H, $E(H) \subseteq E$, with $O(n^{1+\lambda})$ edges, such that for every pair of vertices $u, w \in V$ with $dist_G(u,w) \geq \beta'(\epsilon,\lambda)$, $dist_H(u,w) \leq (1+\epsilon) dist_G(u,w)$.*

Proof. By Theorem 3.17 there exists a constant $\beta(\epsilon/2, \lceil 1/\lambda \rceil)$ such that there exists a polynomial time constructible $(1+\epsilon/2, \beta(\epsilon/2, \lceil 1/\lambda \rceil))$-spanner H, $E(H) \subseteq E$, with $O(n^{1+\lambda})$ edges. Set $\beta'(\epsilon,\lambda) = 2\beta(\epsilon/2, \lceil 1/\lambda \rceil) \,/\, \epsilon$ and observe that for any pair of vertices $u, w \in V$ with $dist_G(u,w) \geq \beta'(\epsilon,\lambda)$,

$$dist_H(u,w) \;\leq\; (1+\epsilon/2) dist_G(u,w) + \beta(\epsilon/2, \lceil 1/\lambda \rceil) \;\leq\; (1+\epsilon) dist_G(u,w)\,. \qquad \blacksquare$$

In order to get a multiplicative stretch factor asymptotically close to 1 we just substitute $\epsilon = 1/\log^b n$ for any constant b in Theorem 3.17.

COROLLARY 3.19. *For any fixed integer $2 \leq \kappa = O(\log n)$ and constant $b > 0$ and for any graph G, running Algorithm* SP_CONS *with G, κ, $\beta(\kappa,n) = \kappa^{\log\log \kappa} \log^{b \log \kappa} n$ and $J = \log \kappa$ yields a $(1 + 1/\log^b n, \beta(\kappa,n))$-spanner of G with $O(\beta(\kappa,n) \cdot n^{1+1/\kappa})$ edges.*

Again, for any constant $\kappa \geq 2$ this yields a $(1 + 1/\operatorname{polylog} n, \operatorname{polylog} n)$-spanner with $\tilde{O}(n^{1+1/\kappa})$ edges.

In order to get an almost linear number of edges (or formally, $o(n^{1+\nu})$ for any $\nu > 0$) we substitute $\epsilon = (\log n)^{\log^{(3)} n}$ and $\kappa = \log n$ in Theorem 3.17, and get the following.

COROLLARY 3.20. *For any graph G, running Algorithm* SP_CONS *with G, $\kappa = \log n$, $\beta(n) = (\log n)^{O(\log^{(2)} n \log^{(3)} n)}$ and $J = \log \kappa$ yields a $(1+1/(\log n)^{\log^{(3)} n}, \beta(n))$-spanner of G with $O(\beta(n)n)$ edges.*

Theorem 3.17 also enables us to decrease the multiplicative stretch to $1 + 2^{-\log n/\log^{(b)} n}$ while not significantly increasing the other parameters. Specifically, we obtain the following.

COROLLARY 3.21. *For any fixed $2 \leq \kappa = O(\log n)$, constant $b \geq 2$ and graph G, running Algorithm* SP_CONS *with G, κ, $J = \log \kappa$ and $\beta(\kappa,n) = \kappa^{\log\log \kappa} \cdot 2^{\log \kappa \log n/\log^{(b)} n}$ yields a $(1 + 2^{-\log n/\log^{(b)} n}, \beta(\kappa,n))$-spanner of G with $O(\beta(\kappa, n)n^{1+\frac{1}{\kappa}})$ edges.*

In particular, for $\kappa = \log n$ this yields a $(1 + 2^{-\log n/\log^{(b)} n}, 2^{O(\log n/\log^{(b)} n)})$-spanner with $2^{O(\log n/\log^{(b)} n)} n$ edges.

Finally, we remark that for some specific small values of κ, tighter bounds on ϵ and β parameters of the spanner that is constructed by Algorithm SP_CONS, can be obtained (see the preliminary versions of this paper [9, 10] for the details). In particular, the following statements hold.

THEOREM 3.22.

1. *Algorithm* SP_CONS*, when run with parameter* $\kappa = 2$*, yields a construction of an additive* 2*-spanner with* $O(n^{3/2})$ *edges.*
2. *Algorithm* SP_CONS*, when run with parameter* $\kappa = 3$*, yields a construction of a* $(1+\epsilon, 4)$*-spanner with* $O(\epsilon^{-1}n^{4/3})$ *edges for any* $\epsilon > 0$*.*

The first result improves by a logarithmic factor the result of [7] and it is tight up to constant factors due to an $\Omega(n^{3/2})$ lower bound of [16]. However, the running time of Algorithm SP_CONS with parameter $\kappa = 2$ is $O(n^{5/2})$, which is significantly larger than the running time of the algorithm of [7], which is $\tilde{O}(n^2)$. The second result significantly improves the previously known construction of 5-spanner with $O(n^{4/3})$ edges, due to [1].

4. Running time. In this section we analyze the running time of Algorithm SP_CONS. We then show that it can be modified to run in time $\tilde{O}(n^{2+\mu})$, for arbitrarily small $\mu > 0$, while maintaining α, β and the size of the generated spanner as before.

4.1. Time complexity of Algorithm SP_CONS. We start by analyzing the running time of Algorithm SP_CONS as described above, without modifications.

LEMMA 4.1. *The running time of Procedure* DOWN_PART *is* $O(|E|+n^{1+1/\kappa}\log n)$.

Proof. Procedure DOWN_PART starts by picking a vertex and running a depth-limited version of the unweighted BFS algorithm from this vertex. The BFS algorithm continues adding new layers until the next iteration increases the size of the cluster by a factor smaller than $n^{1/\kappa}$. Let S be some cluster of the spanned partition $\mathcal{G}$ built by the procedure. By using appropriate data structures, the operation of counting the number of vertices at the next layer that joins the cluster takes $O(|S|n^{1/\kappa}\log(|S|n^{1/\kappa}))$ time, since the total number of vertices involved in this process is $O(|S|n^{1/\kappa})$. Hence summing over all the cluster constructions, the running time invested in deciding whether to terminate the construction of the current cluster and to start the construction of the next cluster is

$$\sum_{S\in\mathcal{G}} O(|S|n^{1/\kappa}\log(|S|n^{1/\kappa})) \;=\; O\left(n^{1/\kappa}\log n\sum_{S\in\mathcal{G}}|S|\right) \;=\; O(\log n\cdot n^{1+1/\kappa})\,.$$

Note that when constructing a new cluster, Procedure DOWN_PART might touch the vertices of the shell of an already built cluster S, but it will never explore again the edges that have at least one endpoint in S. Note also that the edges connecting two vertices of the shell of S were not explored during the construction of the cluster S. Thus Procedure DOWN_PART never re-explores edges that were previously explored.

For a cluster $S_i \in \mathcal{G}$, let $E'(S_i)$ be the set of edges explored during the construction of the cluster S_i. By the above considerations, $E'(S_i)\cap E'(S_j) = \emptyset$ for any two clusters $S_i, S_j \in \mathcal{G}$. Also $\cup_{S\in\mathcal{G}}E'(S) \subseteq E$. The depth-limited unweighted BFS algorithm that explores m' edges and n' vertices requires $O(m' + n'\log n')$ time. Hence the total running time of all invocations of the depth-limited unweighted BFS algorithm by Procedure DOWN_PART is $\sum_{S\in\mathcal{G}}|E'(S)| + O(n^{1+1/\kappa}\log n) = O(|E| + n^{1+1/\kappa}\log n)$. Hence the overall running time of the procedure is $O(|E| + n^{1+1/\kappa}\log n)$. ∎

Note that the problem is interesting mainly when $|E|$ is greater than $O(n^{1+1/\kappa})$, since otherwise the graph itself may serve as its own sparse 1-spanner, with $O(n^{1+1/\kappa})$ edges.

LEMMA 4.2. *Procedure* SC *can be executed on an input spanned partition* $\mathcal{C}$ *in* $O(|E|n/\check{S}(\mathcal{C}))$ *time.*

Proof. It is easy to see that the most expensive parts of Procedure SC are steps 2(b)iiA and 4, which are concerned with computing the shortest paths between clusters. This task can be performed by running BFS algorithm separately from every cluster. Note that BFS algorithm enables computing all the distances from a single source, where this source need not be a single vertex but may be a subset of vertices as well. Recall that for a subset of vertices U and a vertex v, the distance between U and v is $\min\{dist_G(u,v) \mid u \in U\}$. Each invocation of BFS algorithm requires $O(|E| + n \cdot \log n) = O(|E|)$ time. The number of clusters is $O(n/\check{S}(\mathcal{C}))$. Hence the running time of the procedure is $O(|E|n/\check{S}(\mathcal{C}))$. ∎

LEMMA 4.3. *The running time of Algorithm* SP_CONS *is* $O(|E|n^{(\kappa-1)/\kappa})$.

Proof. The dominant term in the time complexity of the algorithm is the running time of the very first invocation of Procedure SC. In this invocation, $\check{S}(\mathcal{G}) \geq n^{1/\kappa}$, and so by Lemma 4.2 it requires $O(|E|n^{(\kappa-1)/\kappa})$ time. The running time of all later invocations of SC is significantly smaller, as those are applied to spanned partitions $\mathcal{C}$ with $\check{S}(\mathcal{C}) \geq n^{2/\kappa}$. The complexity of step 5 can be analyzed along the same lines as in the proof of Lemma 4.2 and shown to be at most $O(|E|n^{1/\kappa})$. The running time of Procedure DOWN_PART is at most $O(|E| + n^{1+1/\kappa} \log n)$. Thus, the overall complexity of Algorithm SP_CONS is $O(|E|n^{(\kappa-1)/\kappa})$. ∎

4.2. Speeding up algorithm SP_CONS. In this section we present a modification of Algorithm SP_CONS that has smaller time complexity, but α, β and size parameters similar to those of section 3.3.

The main idea is to save time by finding *almost* shortest paths instead of shortest ones. Specifically, the modified algorithm receives an additional parameter $t \geq 1$. It starts with invoking an *all pairs almost shortest path* algorithm $APASP_t$ due to [7]. This algorithm runs for $\tilde{O}(n^{2+1/t})$ time and for every pair of vertices $u, w \in V$ computes a path of length $dist'(u,w) \leq dist_G(u,w) + t$. Next, the algorithm sorts the obtained n^2 distances in $O(n^2 \log n)$ time and for each pair maintains the index in the sorted array (i.e., a pair of closer vertices will have a smaller index). Next, it runs Algorithm SP_CONS as is, except that whenever it needs to compute a shortest path between two clusters it uses the precomputed array. Specifically, for two clusters S_i and S_j it takes $|S_i| \cdot |S_j|$ time to find the pair of vertices $u_i \in S_i$, $u_j \in S_j$ such that $dist'(u_i,u_j) = \min\{dist'(u,w) \mid u \in S_i, w \in S_j\}$. Hence, overall, the approximate computation of all the distances between clusters of some partition $\mathcal{C}$ and all the paths between close cluster pairs (i.e., at distance bounded by $\Upsilon^{i/J}$ for some $1 \leq i \leq J$) takes time bounded by

$$O\left(\sum_{S_i,S_j \in \mathcal{C}} \Upsilon |S_i| \cdot |S_j|\right) \;\leq\; O\left(\Upsilon \left(\sum_{S_i \in \mathcal{C}} |S_i|\right)^2\right) \;=\; O(\Upsilon n^2)\ .$$

Since there are $O(J)$ iterations, the distances and paths for different partitions are computed $O(J)$ times, i.e., the total running time of computing the distances and paths is $O(J\Upsilon n^2)$. Hence, the overall running time of the modified algorithm is $\tilde{O}(n^{2+1/t} + J\Upsilon n^2)$, and setting $J = \log \kappa = O(\log\log n)$ and $\Upsilon = O(\kappa^{\log \kappa}) = O((\log n)^{\log\log n})$, the resulting time bound is $\tilde{O}(n^{2+1/t})$.

It remains to analyze the α, β and size parameters of the spanner obtained by the above modification of Algorithm SP_CONS. First, instead of Lemma 3.1 we now get

$$\hat{D}(\mathcal{C}') \;\leq\; 3\hat{D}(\mathcal{C}) + 2\delta + 2t. \tag{14}$$

Next, Lemma 3.3 is replaced by the following.

LEMMA 4.4. *For every integer* $1 \leq j \leq J-1$,

$$\text{(a)} \quad \hat{D}(\mathcal{C}'_j) \leq 2\left(3^j \kappa t_{J-1} + \sum_{l=1}^{j} 3^{j-l}\Upsilon^{l/J}\right) + t(3^j - 1) \ ,$$

$$\text{(b)} \quad \hat{D}(\mathcal{R}_j) \leq 2\left(3^{j-1} \kappa t_{J-1} + \sum_{l=1}^{j-1} 3^{j-1-l}\Upsilon^{l/J}\right) + t(3^{j-1} - 1) \ .$$

Proof. The induction base holds since $\hat{D}(\mathcal{R}_1) \leq \hat{D}(\mathcal{C}_1) \leq 2t_{J-1}\kappa + t(3^0 - 1) = 2t_{J-1}\kappa$ and $\hat{D}(\mathcal{C}_1) \leq 2(3^1 t_{J-1}\kappa + \Upsilon^{1/J}) + t(3^1 - 1)$, by Lemma 3.3 and inequality (14).

The induction hypothesis changes to

$$\hat{D}(\mathcal{C}'_j) \ \leq \ 2\left(3^j \kappa t_{J-1} + \sum_{l=1}^{j} 3^{j-l}\Upsilon^{l/J}\right) + t(3^j - 1) \ .$$

The inequalities (4) and (5) are unchanged, and hence

$$\hat{D}(\mathcal{R}_{j+1}) \ \leq \ \hat{D}(\mathcal{C}_{j+1}) \ \leq \ 2\left(3^j \kappa t_{J-1} + \sum_{l=1}^{j} 3^{j-l}\Upsilon^{l/J}\right) + t(3^j - 1) \ .$$

So by inequality (14), it follows that

$$\begin{aligned}\hat{D}(\mathcal{C}_{j+1}) &\leq 3 \cdot 2\left(3^j \kappa t_{J-1} + \sum_{l=1}^{j} 3^{j-l}\Upsilon^{l/J}\right) + 3t(3^j - 1) + 2\Upsilon^{(j+1)/J} + 2t \\ &= 2\left(3^{j+1} \kappa t_{J-1} + \sum_{l=1}^{j+1} 3^{j+1-l}\Upsilon^{l/J}\right) + t(3^{j+1} - 1) \ . \qquad \square\end{aligned}$$

Analogously to inequality (6), it follows that

$$\hat{D}(R_J) \ \leq \ 2\left(3^{J-1} \kappa t_{J-1} + \sum_{l=1}^{J-1} 3^{J-l-1}\Upsilon^{l/J}\right) + t(3^{J-1} - 1) \ .$$

Next, it is easy to see that inequality (7) becomes

$$|E(H_j)| \ = \ O(n^{1+t_{J-j}-2t_{J-j+1}}(\delta_{J+1-j} + t)) \ .$$

Hence

$$|E(H)| \ = \ \sum_{j=1}^{J-1} |E(H_j)| \ = \ O(n^{1+1/\kappa}(\Upsilon + tJ)) \ .$$

Lemmas 3.13 and 3.14 are unchanged. However, since the expression for $\hat{D}(\mathcal{R}_j)$ is modified, inequality (13) becomes

$$\begin{aligned}\gamma_j/4 &\leq 2^{j-1} t_{J-1}\kappa + 2^{j-2}\left(3t_{J-1}\kappa + \Upsilon^{1/J} + 2t\right) + \cdots \\ &\quad + 2^0\left(3^{j-1} t_{J-1}\kappa + \sum_{i=1}^{j-1} 3^{j-1-i}\Upsilon^{i/J} + t(3^{j-1} - 1)\right) \\ &= t_{J-1}\kappa \sum_{i=0}^{j-1} 3^i 2^{j-1-i} + \sum_{l=1}^{j-1}\left(\Upsilon^{l/J} \sum_{i=0}^{j-l-1} 3^i 2^{j-l-i-1}\right) + t\sum_{i=1}^{j-1} 2^{j-1-i}(3^i - 1) \\ &\leq t_{J-1}\kappa 3^j + 6\Upsilon^{(j-1)/J} + t3^j \ = \ 3^j(t_{J-1}\kappa + t) + 6\Upsilon^{(j-1)/J} \ .\end{aligned}$$

Hence the multiplicative factor is at most $1 + 2t_J\kappa + O(\frac{J + t_{J-1}\kappa + t}{\Upsilon^{1/J}})$, the additive term is

$O((t_{J-1}\kappa+t)3^J+\Upsilon^{(J-1)/J})$ and the size of the constructed spanner is $O(n^{1+1/\kappa}(\Upsilon+tJ))$, and we have the following.

THEOREM 4.5. *For any fixed $0<\epsilon<1$ and fixed integers $t\geq 1$ and $\kappa\geq 2$ there exist fixed $\bar{\beta}=\bar{\beta}(\kappa,\epsilon,t)=\kappa^{\max\{\log\log\kappa-\log\epsilon,3\}}+t\log\kappa$ and $\tilde{\beta}=\tilde{\beta}(\kappa,\epsilon,t)=\kappa^{\max\{\log\log\kappa-\log\epsilon,\log t,3\}}$ such that every n-vertex graph has a $(1+\epsilon,\tilde{\beta})$-spanner of size bounded by $O(n^{1+1/\kappa}\bar{\beta})$ that can be built in time $\tilde{O}(n^{2+1/t})$.*

In particular, setting $t=O(\log\kappa)$ we obtain results analogous to Theorem 3.17 and Corollaries 3.18–3.20 by an algorithm of time complexity $\tilde{O}(n^{2+1/\log\kappa})$. Getting a running time of $\tilde{O}(n^{2+1/\kappa})$ requires raising the additive term to $O(\kappa^{\max\{\log\kappa,-\log\epsilon\}})$, which is, nonetheless, still constant for constant κ and ϵ. This enables us to generalize Corollary 3.18 and get the following.

COROLLARY 4.6. *For any constant $\epsilon>0,\lambda>0,\mu>0$ there exists a constant $\beta''(\epsilon,\lambda,\mu)$ such that for any n-vertex graph $G=(V,E)$ there exists a subgraph H, $E(H)\subseteq E$, that satisfies the following properties*

1. *For every pair of vertices $u,w\in V$ with $dist_G(u,w)\geq\beta''(\epsilon,\lambda,\mu)$, $dist_H(u,w)\leq(1+\epsilon)dist_G(u,w)$.*
2. *$|E(H)|=O(n^{1+\lambda})$.*
3. *H can be constructed in $\tilde{O}(n^{2+\mu})$ time.*

Acknowledgments. We thank Uri Zwick for his helpful comments, corrections and suggestions of improvements, and an anonymous referee for helpful remarks.

REFERENCES

[1] I. ALTHÖFER, G. DAS, D. DOBKIN, D. JOSEPH, AND J. SOARES, *On sparse spanners of weighted graphs*, Discrete Comput. Geom., 9 (1993), pp. 81–100.
[2] B. AWERBUCH AND D. PELEG, *Sparse partitions*, in Proceedings of the 31st Annual IEEE Symposium on Foundations of Computer Science, 1990, pp. 503–513.
[3] B. BOLLOBAS, D. COPPERSMITH, AND M. L. ELKIN, *Sparse subgraphs that preserve long distances and additive spanners*, in Proceedings of the 14th Annual ACM-SIAM Symposium on Discrete Algorithms, Baltimore, MD, 2003.
[4] B. CHANDRA, G. DAS, G. NARASIMHAN, AND J. SOARES, *New sparseness results on graph spanners*, in Proceedings of the 8th Annual ACM Symposium on Computational Geometry, Berlin, 1992, pp. 192–201.
[5] L. P. CHEW, *There is a planar graph almost as good as the complete graph*, in Proceedings of the 2nd Annual Symposium on Computational Geometry, 1986, pp. 169–177.
[6] D. P. DOBKIN, S. J. FRIEDMAN, AND K. J. SUPOWIT, *Delaunay graphs are almost as good as complete graphs*, in Proceedings of the 31st Annual IEEE Symposium on Foundations of Computer Science, 1987, pp. 20–26.
[7] D. DOR, S. HALPERIN, AND U. ZWICK, *All pairs almost shortest paths*, SIAM J. Comput., 29 (2000), pp. 1740–1759.
[8] M. L. ELKIN, *Computing Almost Shortest Paths*, in Proceedings of the 20th Annual ACM Symposium on Principles of Distributed Computing, Newport, RI, 2001, pp. 53–63.
[9] M. L. ELKIN AND D. PELEG, *$(1+\epsilon,\beta)$-Spanner Constructions for General Graphs*, Technical Report MCS00-17, Weizmann Institute of Science, Rehovot, Israel, 2000.
[10] M. L. ELKIN AND D. PELEG, *$(1+\epsilon,\beta)$-Spanner constructions for general graphs*, in Proceedings of the 33rd Annual ACM Symposium on Theory of Computing, Crete, Greece, 2001.
[11] A. L. LIESTMAN AND T. C. SHERMER, *Additive Graph Spanners*, Tech. Report 91-5, Simon Fraser University, Burnaby, BC, Canada, 1991.
[12] A. L. LIESTMAN AND T. C. SHERMER, *Grid spanners*, Networks, 23 (1993), pp. 123–133.
[13] D. PELEG, *Distributed Computing: A Locality-Sensitive Approach*, SIAM, Philadelphia, PA, 2000.
[14] D. PELEG AND A. SCHÄFFER, *Graph spanners*, J. Graph Theory, 13 (1989), pp. 99–116.
[15] D. PELEG AND J. D. ULLMAN, *An optimal synchronizer for the hypercube*, SIAM J. Comput., 18 (1989), pp. 740–747.
[16] R. WENGER, *Extremal graphs with no C^4's, C^6's and C^{10}'s*, J. Combin. Theory Ser. B, 52 (1991), pp. 113–116.