Abstract  For an unweighted undirected graph $G = (V, E)$, and a pair of positive integers $\alpha \geq 1$, $\beta \geq 0$, a subgraph $G' = (V, H)$, $H \subseteq E$, is called an $(\alpha, \beta)$-spanner of $G$ if for every pair of vertices $u, v \in V$, $\text{dist}_{G'}(u, v) \leq \alpha \cdot \text{dist}_G(u, v) + \beta$.

It was shown in [21] that for any $\epsilon > 0$, $\kappa = 1, 2, \ldots$, there exists an integer $\beta = \beta(\epsilon, \kappa)$ such that for every $n$-vertex graph $G$ there exists a $(1 + \epsilon, \beta)$-spanner $G'$ with $O(n^{1+1/\kappa})$ edges. An efficient distributed protocol for constructing $(1 + \epsilon, \beta)$-spanners was devised in [19]. The running time and the communication complexity of that protocol are $O(n^{1+\rho})$ and $O(\lvert E \rvert n^\rho)$, respectively, where $\rho$ is an additional control parameter of the protocol that affects only the additive term $\beta$.

In this paper we devise a protocol with a drastically improved running time ($O(n^\rho)$ as opposed to $O(n^{1+\rho})$) for constructing $(1 + \epsilon, \beta)$-spanners. Our protocol has the same communication complexity as the protocol of [19], and it constructs spanners with essentially the same properties as the spanners that are constructed by the protocol of [19]. The protocol can be easily extended to a parallel implementation which runs in $O(\log n + (\lvert E \rvert \cdot n^\rho \log n)/p)$ time using $p$ processors in the EREW PRAM model. In particular, when the number of processors, $p$, is at least $\lvert E \rvert \cdot n^\rho$, the running time of the algorithm is $O(\log n)$. We also show that our protocol for constructing $(1 + \epsilon, \beta)$-spanners can be adapted to the streaming model, and devise a streaming algorithm that uses a constant number of passes and $O(n^{1+1/\kappa} \cdot \log n)$ bits of space for computing all-pairs-almost-shortest-paths of length at most by a multiplicative factor $(1 + \epsilon)$ and an additive term of $\beta$ greater than the shortest paths. Our algorithm processes each edge in time $O(n^\rho)$, for an arbitrarily small $\rho > 0$. The only previously known algorithm for the problem [23] constructs paths of length $\kappa$ times greater than the shortest paths, has the same space requirements as our algorithm, but requires $O(n^{1+1/\kappa})$ time for processing each edge of the input graph. However, the algorithm of [23] uses just one pass over the input, as opposed to the constant number of passes in our algorithm.\footnote{After the preliminary version of our paper [22] appeared on PODC’04, Feigenbaum et al. [24] came up with a new streaming algorithm for the problem that is far more efficient than [23] in terms of time-per-edge processing. However, our algorithm is still the only existing streaming algorithm that provides an almost additive approximation of distances.}

We also show that any streaming algorithm for $o(n)$-approximate distance computation requires $\Omega(n)$ bits of space.

Keywords  Spanner · Almost shortest paths · Streaming model

1 Introduction

In this paper we study the problem of constructing spanners and computing almost shortest paths in the distributed and streaming models of computation.

1.1 Distributed model

Consider an unweighted undirected graph $G = (V, E)$. Observe that the graph $G$ induces a metric space $U$ in which the vertex set $V$ serves as the set of points, and the lengths of the shortest paths serve as distances. Intuitively, a graph $G' = (V, H)$, $H \subseteq E$ is a sparse skeleton of the
graph $G$ whose induced metric space $\mathcal{U}'$ is a close approximation of the metric space $\mathcal{U}$ of the graph $G$. Graph spanners have multiple applications in the areas of Graph Algorithms and Distributed Computing, and were subject of extensive research [4, 9, 10, 11, 13, 18, 19, 21, 33, 36] in the course of the last fifteen years. In particular, spanners were used for routing [7, 35], for constructing synchronizers [6, 34], and for computing almost shortest paths [4, 19].

In the area of Distributed Computing spanners were found particularly useful in the following (henceforth called distributed) model of computation. In this model every vertex of an $n$-vertex graph $G = (V, E)$ hosts a processor with an unbounded computational power but only limited knowledge. Specifically, it is assumed that in the beginning of the computation every processor $v$ knows only the identities of its neighbors. The communication is synchronous and proceeds in discrete pulses, called rounds. On each round each processor is allowed to send short (possibly different) messages to all its neighbors. The (worst-case) running time of a distributed algorithm is the (worst-case) number of rounds that are required for the algorithm to complete its execution. The design of distributed algorithms is a vivid area of research (see, e.g., [32], and the references therein). Spanners serve as an important tool in this area, and particularly, they were used for routing [13, 34], for constructing synchronizers [6, 34], and for computing almost shortest paths [19].

For all these applications it is crucially important that spanner is a part of the original network, and consequently, the processors can communicate over its links. In particular, the processors can execute over the links of a spanner any protocol that was designed for arbitrary networks. Also, since a spanner approximates the distances of the original network, the execution of a protocol on a spanner is almost as time-efficient as its execution on the original network (spanned by the spanner). However, since spanners are typically much sparser than the networks they span, an execution of a protocol on a spanner is typically much more communication-efficient than the corresponding execution on the original network. These properties make spanners extremely valuable in the design of distributed protocols, and raise the problem of designing efficient distributed protocols for constructing spanners with good parameters. In this paper we address this problem.

In the '90s most of the study of spanners and their applications focused on spanners whose metric space distorts the original metric space by at most a constant multiplicative factor. More formally, for a positive integer $\kappa = 1, 2, \ldots$, a subgraph $G' = (V, H)$ is a $\kappa$-spanner of the graph $G = (V, E)$, if for every pair of vertices $u, v \in V$, $dist_{G'}(u, v) \leq \kappa \cdot dist_G(u, v)$ (where $dist_G(u, v)$ stands for the distance between the vertices $u$ and $v$ in the graph $G$). A fundamental theorem concerning $\kappa$-spanners, that was proven by Peleg and Schäffer [33], says that for every $n$-vertex graph $G = (V, E)$ and a positive integer $\kappa = 1, 2, \ldots$, there exists a $\kappa$-spanner with $n^{1+O(\frac{1}{\kappa})}$ edges, and that this is the best possible up to the constant hidden by the $O$-notation.

More recently, Elkin and Peleg studied a more general notion of $(\alpha, \beta)$-spanner: a subgraph $G' = (V, H)$ of the graph $G = (V, E)$ is an $(\alpha, \beta)$-spanner of the graph $G$ if for every pair of vertices $u, v \in V$, $dist_{G'}(u, v) \leq \alpha \cdot dist_G(u, v) + \beta$. They have shown that for every $n$-vertex graph $G = (V, E)$, there exists a $(1 + \epsilon, \beta)$-spanner $G' = (V, H)$ of $G$ with $O(n^{1+1/\kappa})$ edges. This result shows that the tradeoff of Peleg and Schäffer [33] can be drastically improved if one is concerned only with approximating the distances that are larger than a certain constant.

While the proof of Elkin and Peleg [21] is not known to translate to an efficient distributed algorithm, soon afterwards Elkin [19] came up with an alternative proof of this theorem, which though providing somewhat inferior constants, translates directly into efficient distributed and sequential algorithms. The latter algorithms enabled [19] to use $(1 + \epsilon, \beta)$-spanners for efficient algorithms for computing almost shortest paths from $s$ sources.

Specifically, it was shown in [19] that for every $n$-vertex graph $G = (V, E)$, positive integer $\kappa = 1, 2, \ldots$, and positive numbers $\epsilon, \rho > 0$, there exists a distributed algorithm that constructs $(1 + \epsilon, \beta)$-spanners with $O(n^{1+1/\kappa})$ edges in time $O(n^{1+\rho})$ with message complexity $O(|E| \cdot n^\rho)$. Note that while the message complexity of this result is near-optimal, its running time is prohibitively large. In this paper we drastically improve this running time and devise a randomized distributed algorithm that constructs $(1 + \epsilon, \beta)$-spanners with $O(n^{1+1/\kappa})$ edges in time $O(n^\rho)$, and with message complexity $O(|E| \cdot n^\rho)$. Note that the message complexity of our algorithm is no worse than the message complexity of the algorithm of [19], and the parameters of the obtained spanners are essentially the same as in the result of [19]. This result directly translates to an improved distributed algorithm for computing almost shortest paths from $s$ sources.

We remark that both our algorithm and the algorithm of [19] can be adapted to the asynchronous model of distributed computation in a rather straightforward way by using synchronizers of [5]. The parameters of the obtained asynchronous algorithms are essentially the same as the parameters of the synchronous algorithms. Furthermore, our algorithm can also be easily extended to a parallel implementation which runs in $O(log n + (|E| \cdot n^\rho \log n) / p)$ time using $p$ processors in the EREW PRAM model. Particularly, when the number of processors, $p$, is at least $|E| \cdot n^\rho$, the running time of the algorithm is $O(log n)$. This is the first known parallel algorithm for constructing sparse $(1 + \epsilon, \beta)$-spanners.

### 1.2 Streaming model

We also adapt our distributed algorithm for constructing spanners to the streaming model. In this model, the computation is centralized, and is performed by a single processor. However, unlike the traditional computational models, in the streaming model the processor is not allowed to store