Distributed Computing
https://doi.org/10.1007/s00446-018-0326-6

# Distributed distance computation and routing with small messages

**Christoph Lenzen**[1] · **Boaz Patt-Shamir**[2] · **David Peleg**[3]

Received: 7 February 2017 / Accepted: 3 February 2018


**Abstract**
We consider shortest paths computation and related tasks from the viewpoint of network algorithms, where the $n$-node input graph is also the computational system: nodes represent processors and edges represent communication links, which can in each time step carry an $\mathcal{O}(\log n)$-bit message. We identify several basic distributed distance computation tasks that are highly useful in the design of more sophisticated algorithms and provide efficient solutions. We showcase the utility of these tools by means of several applications.

**Keywords** CONGEST model · Source detection · Skeleton spanner · Compact routing · All-pairs shortest paths · Single-souce shortest paths

## 1 Introduction

The task of routing table construction concerns computing local tables at all nodes of a network that will allow each node $v$, when given a destination node $u$, to instantly find the first link on a route from $v$ to $u$, from which the next hop is found by another lookup etc. Constructing routing tables is a central task in network operation, the Internet being a prime example. Routing table construction (abbreviated RTC henceforth) is not only important as an end goal, but is also a critical part of the infrastructure in most distributed systems.

At the heart of any routing protocol lies the computation of short paths between all possible node pairs, which is another fundamental challenge that occurs in a multitude of optimization problems. The best previous distributed algorithms for this task were based on, essentially, running $n$ independent versions of a single-source shortest-paths algorithm, where $n$ is the number of nodes in the network: in each version a different node acts as the source. The result of this approach is an inherent $\Omega(n)$ complexity bottleneck in message size or execution time, and frequently both.

In this work, we provide fundamental building blocks and obtain sub-linear-time distributed algorithms for a variety of distance estimation and routing tasks in the so-called **CONGEST** model. In this model, each node has a unique $\mathcal{O}(\log n)$-bit identifier, and it is assumed that in each time unit, nodes can send and receive, on each of their incident links, messages of $\mathcal{O}(\log n)$ bits, where $n$ denotes the number of nodes in the system. This means that each message can carry no more than a constant number of node identifiers and integers of magnitude polynomial in $n$. Communication proceeds in synchronous rounds and the system is assumed to be fault-free. Initially, nodes know only the identity of their neighbors and, if the graph is weighted, the weights of adjacent edges.

It is quite obvious that many distributed tasks, including RTC, cannot be solved in fewer rounds than the network diameter, because some information needs to cross the entire network. It is also well-known (see, e.g., [15]) that in **CONGEST** model, many basic tasks cannot be solved in $\tilde{o}(\sqrt{n})$

This article is based on preliminary results appearing at conferences [32,34,35]. This work has been supported by the Swiss National Science Foundation (SNSF), the Swiss Society of Friends of the Weizmann Institute of Science, the Deutsche Forschungsgemeinschaft (DFG, reference number Le 3107/1-1), the Israel Science Foundation (Grants 894/09 and 1444/14), the United States-Israel Binational Science Foundation (Grant 2008348), the Israel Ministry of Science and Technology (infrastructures grant), the Citi Foundation, and the I-CORE program of the Israel PBC and ISF (Grant 4/11).

✉ Christoph Lenzen
clenzen@mpi-inf.mpg.de

[1] MPI for Informatics, Campus E1.4, 66123 Saarbrücken, Germany

[2] School of Electrical Engineering, Tel Aviv University, 69978 Tel Aviv, Israel

[3] Faculty of Mathematics and Computer Science, Weizmann Institute of Science, 76100 Rehovot, Israel

Published online: 13 February 2018

rounds in some graphs with very small diameter.[1] As we show, such a lower bound extends naturally to RTC and other related tasks. We provide algorithms whose running time is close to the lower bound.

## 1.1 Main contributions

While the derivation of the results on routing and distance approximation requires overcoming non-trivial technical challenges, the main insight we seek to convey in this article is the identification of a few fundamental tasks whose efficient solution facilitates fast distributed algorithms. These basic tasks include what we call *exact* and *approximate source detection*, and *skeleton spanner* construction. For each of these tasks, we provide an optimal or near-optimal distributed implementation, which in turn results in a variety of (nearly) optimal solutions to distance approximation, routing, and similar problems. Let us specify what these tasks are.

### 1.1.1 Source detection

Intuitively, in the source detection problem there is a subset $S$ of nodes called sources, and a parameter $\sigma \in \mathbb{N}$. The required output at each node is a list of its $\sigma$ closest sources, alongside the respective distances. This is a very powerful basic routine, as it generalizes various distance computation and breadth-first-search (BFS) tree construction problems. For instance, the all-pairs shortest path problem (APSP) can be rephrased as source detection with $S = V$ and $\sigma = |V|$ (where $V$ is the set of all nodes), and single-source shortest paths translates to $|S| = \sigma = 1$.

For the general case of $\sigma < |S|$, however, this intuitive description must be refined. Source detection implies construction of *partial* BFS trees rooted at the nodes in $S$, where each node participates in the trees rooted at its closest $\sigma$ sources. To ensure that the parent of a node in the shortest-paths tree rooted at $s \in S$ also has $s$ in its list, we impose consistent tie-breaking, by relying on the unique node identifiers (any other consistent tie-breaking mechanism could do as well).

A second salient point is that we limit the "horizon", namely the number of hops up to which sources are considered, because determining distances may require communication over $|V| - 1$ hops in the worst case. By bounding both the number of sources to detect and the hop count up to which this is required, we avoid trivial $\Omega(n)$ lower bounds on the running time. With these issues in mind, the source detection problem on unweighted graphs is formalized as follows.

[1] We use weak asymptotic notation throughout the paper, where $\tilde{\mathcal{O}}$, $\tilde{\Omega}$, etc. absorb polylog $n$ factors (irrespective of the considered function, e.g., $\tilde{\mathcal{O}}(1) = (\log n)^{\mathcal{O}(1)}$), where $n$ is the number of nodes in the graph.

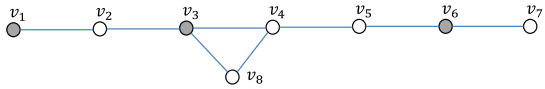


**Fig. 1** An example of unweighted source detection. Shaded nodes represent sources. For $\sigma = 2$ and $h = 3$ and assuming $v_i < v_j$ for $i < j$ we have, for example, the outputs $L_{v_2} = \langle (1, v_1), (1, v_3) \rangle$, $L_{v_7} = \langle (1, v_6) \rangle$ and $L_{v_8} = \langle (1, v_3), (3, v_1) \rangle$

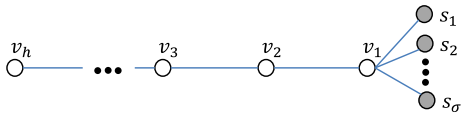


**Fig. 2** A graph where uweighted source detection must take at least $h + \Omega(\sigma)$ rounds. The shaded nodes $s_1 \ldots, s_\sigma$ are sources. Node $v_h$ receives the first record of a source after $h$ rounds. Note that if only one distance/source pair fits into a message, the bound becomes precisely $h + \sigma - 1$

**Unweighted source detection** Fix a graph $G = (V, E)$, and let $\mathrm{hd}(v, w)$ denote the distance between any two nodes $v, w \in V$. (We use hd() to emphasize that this distance is measured in terms of *hops*.) Let $\mathbb{N}_0$ denote the set of non-negative integers. Let $\mathrm{top}_k(L)$ denote the list of the first $k$ elements of a list $L$, or $L$ if $|L| \leq k$.

**Definition 1.1** (*Unweighted* $(S, h, \sigma)$*-detection*) Given $S \subseteq V$, $v \in V$, and $h \in \mathbb{N}_0$, let $\mathcal{L}_v^{(h)}$ be the list of pairs $\{(\mathrm{hd}(v, s), s) \mid s \in S,\ \mathrm{hd}(v, s) \leq h\}$, ordered in increasing lexicographical order. I.e., $(\mathrm{hd}(v, s), s) < (\mathrm{hd}(v, s'), s')$ iff $\mathrm{hd}(v, s) < \mathrm{hd}(v, s')$, or both $\mathrm{hd}(v, s) = \mathrm{hd}(v, s')$ and the identifiers satisfy $s < s'$.

For $\sigma \in \mathbb{N}$, $(S, h, \sigma)$*-detection* requires each node $v \in V$ to compute $\mathrm{top}_\sigma(\mathcal{L}_v^{(h)})$.

Note that $\sigma$ and/or $h$ may depend on $n$ here; we do not restrict to constant values only.

Figure 1 depicts a simple graph and the resulting lists. We will show that unweighted source detection allows for a fully "pipelined" version of the Bellman–Ford algorithm, running in $\sigma + h - 1$ rounds.

**Theorem 1.2** *Unweighted* $(S, h, \sigma)$*-detection can be solved in* $\sigma + h - 1$ *rounds.*

Given that in our model messages have $\mathcal{O}(\log n)$ bits, only a constant number of source/distance pairs fits into a message. As possibly $\sigma$ such pairs must be sent over the same edge, the above running time is essentially optimal (cf. Fig. 2).

**Weighted source detection** In a weighted graph $G = (V, E, W)$, the situation is more complex. As mentioned above, determining the exact distance between nodes may require tracing a path of $\Omega(n)$ hops. Since we are interested in $o(n)$-time solutions, we relax the requirement of *exact* distances. We use the following notation. Given nodes $v, w \in V$, let $\mathrm{wd}(v, w)$ denote the *weighted* distance between them,

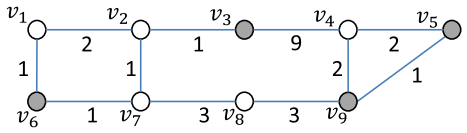


**Fig. 3** A graph where $(S, h+1, \sigma)$-detection cannot be solved in $o(h\sigma)$ rounds. Edge weights are $4ih$ for edges $\{v_i, s_{i,j}\}$ for all $i \in \{1, \ldots, h\}$ and $j \in \{1, \ldots, \sigma\}$, and 1 (i.e., negligible) for all other edges. Node $u_i$, $i \in \{1, \ldots, h\}$, needs to learn about all nodes $s_{i,j}$ and distances $\mathrm{wd}_{h+1}(u_i, s_{i,j})$, where $j \in \{1, \ldots, \sigma\}$. Hence all this information must traverse the dashed edge $\{u_1, v_h\}$. (The example can be modified into one where there are only $\sigma$ sources, each connected to all the $v_i$ nodes. It can be shown, by setting the weight of the edges $\{v_i, s_j\}$ appropriately, that $\sigma h$ values must be communicated over the dashed edge in this case too. Therefore, the special case of $\sigma = |S|$ is not easier.)

and let $\mathrm{wd}_h(v, w)$, called the *h-hop* $v - w$ *distance*, be the weight of the lightest $v - w$ path with at most $h$ edges ($\mathrm{wd}_h(v, w) = \infty$ if no such path exists). We remark that $\mathrm{wd}_h$ is not a metric, since if there is a $v - w$ path of $\ell$ hops with weight less than $\mathrm{wd}_h(v, w)$, then the triangle inequality is violated if $h < \ell \leq 2h$.

**Definition 1.3** (*$(S, h, \sigma)$-detection*) Given $S \subseteq V$, $v \in V$, and $h \in \mathbb{N}_0$, let $\mathcal{L}_v^{(h)}$ be the list of pairs $\{(\mathrm{wd}_h(v, s), s) \mid s \in S,\ \mathrm{wd}_h(v, s) < \infty\}$, ordered in increasing lexicographical order. For $\sigma \in \mathbb{N}$, $(S, h, \sigma)$-*detection* requires each node $v \in V$ to compute $\mathrm{top}_\sigma(\mathcal{L}_v^{(h)})$.

Note that Definition 1.3 generalizes Definition 1.1, as can be seen by assigning unit weight to the edges of an unweighted graph.

Unfortunately, there are instances of the weighted $(S, h, \sigma)$-detection problem that require $\Omega(\sigma h)$ rounds to be solved, as demonstrated by the example given in Fig. 3. The $\mathcal{O}(\sigma h)$ round complexity is easily attained by another variant of Bellman–Ford, where in each iteration, current lists are sent to neighbors, merged and truncated [14,32]. In conjunction with suitable sparsification techniques, this can still lead to algorithms of running time $\tilde{o}(n)$, e.g. for APSP [32]. However, it turns out that relaxing the source detection problem further enables an $\tilde{\mathcal{O}}(\sigma + h)$-round solution and, consequently, better algorithms for APSP and related tasks.

### 1.1.2 Approximate source detection

We relax Definition 1.3 to allow for approximate distances as follows.

**Definition 1.4** (*Approximate Source Detection*) Given $S \subseteq V, h, \sigma \in \mathbb{N}$, and $\varepsilon > 0$, let $\mathcal{L}_v^{(h,\varepsilon)}$ be a list of $\{(\mathrm{wd}'(v, s), s) \mid s \in S,\ \mathrm{wd}'(v, s) < \infty\}$, ordered in increasing lexicographical order, for some $\mathrm{wd}' : V \times S \to \mathbb{N} \cup \{\infty\}$ that satisfies $\mathrm{wd}'(v, s) \in \left[\mathrm{wd}(v, s),\ (1+\varepsilon)\mathrm{wd}_h(v, s)\right]$ for all $v, s \in V$. The $(1+\varepsilon)$-*approximate* $(S, h, \sigma)$-*detection* problem is to output $\mathrm{top}_\sigma(\mathcal{L}_v^{(h,\varepsilon)})$ at each node $v$ for some such $\mathrm{wd}'$.

See Fig. 4 for an example. We stress that we impose very little structure on $\mathrm{wd}'$. In particular,

- $\mathrm{wd}'$ is not required to be a metric (just as $\mathrm{wd}_h$ is not necessarily a metric);
- $\mathrm{wd}'$ is not required to be monotone in $h$ (unlike $\mathrm{wd}_h$);
- $\mathrm{wd}'$ is not required to be symmetric (also unlike $\mathrm{wd}_h$); and
- the list $\mathcal{L}_v^{(h,\varepsilon)}$ could contain entries $(\mathrm{wd}'(v, s), s)$ with $\mathrm{wd}_h(v, s) = \infty$, i.e., $\mathrm{hd}(v, s) > h$.

Unlike for exact source detection, this entails that there is no guarantee that the computed lists induce (approximate, partial) shortest-path trees. In general, this might pose an obstacle to routing algorithms, which tend to exploit such trees. Fortunately, our algorithm for solving approximate source detection is based on solving a number of instances of unweighted source detection, whose solutions provide sufficient information for routing. Assuming positive integer edge weights that are polynomially bounded in $n$, our approach results in a (timewise) near-optimal solution.

**Theorem 1.5** *If $W(e) \in \{1, \ldots, n^\gamma\}$ for all $e \in E$, for a known constant $\gamma > 0$, and $0 < \varepsilon \in \mathcal{O}(1)$, then $(1+\varepsilon)$-approximate $(S, h, \sigma)$-detection can be solved in $\tilde{\mathcal{O}}(\varepsilon^{-1}\sigma + \varepsilon^{-2}h)$ rounds.*

### 1.1.3 Skeleton spanners

When applying source detection as a subroutine, sparsification techniques can help in keeping $\sigma$ small. However, as mentioned above, in weighted graphs it may happen that paths that are shortest in terms of weight have many hops. This difficulty is overcome by constructing a sparse "backbone" of the graph that approximately preserves distances between the nodes of a *skeleton* $\mathcal{S} \subset V$, where $|\mathcal{S}| \in \tilde{\Theta}(\sqrt{n})$. Letting $\mathcal{S}$ be a random sample of nodes of that size, long paths are broken down into subpaths of $\tilde{\mathcal{O}}(\sqrt{n})$ hops between skeleton nodes with high probability.[2] Having information about the distances of skeleton nodes hence usually implies that we can keep $h \in \tilde{\mathcal{O}}(\sqrt{n})$ when applying source detection.

A *skeleton spanner* can be used to concisely represent the global distance structure of the graph and make it available to all nodes in a number of rounds comparable to the lower bound of $\tilde{\Omega}(\sqrt{n} + D)$. Let us formalize this concept. First, we define the *skeleton graph*.

**Definition 1.6** (*Skeleton Graph*) Let $G = (V, E, W)$ be a weighted graph. Given $\mathcal{S} \subseteq V$ and $h \in \mathbb{N}$, the *h-hop $\mathcal{S}$- skeleton graph* is the weighted graph $G_{\mathcal{S},h} = (\mathcal{S}, E_{\mathcal{S},h}, W_{\mathcal{S},h})$ defined by

[2] We use the phrase "with high probability", abbreviated "w.h.p.", as a shorthand for "with probability at least $1 - n^{-c}$, for any desired constant $c$".

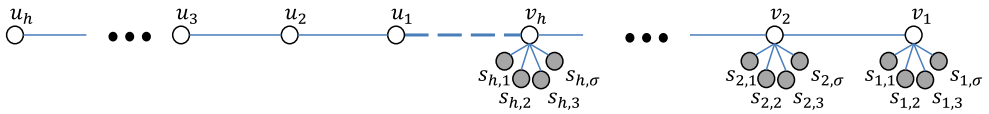


**Fig. 4** An example of approximate source detection, where shaded nodes represent sources. For $\sigma = 4$, $h = 2$, and $\varepsilon = 1/10$ we may have, e.g., $L_{v_3} = \langle (0, v_3), (11, v_5), (12, v_9) \rangle$ and $L_{v_9} = \langle (0, v_9), (1, v_5), (11, v_3), (35, v_6) \rangle$. Note that $12 = \mathrm{wd}'(v_3, v_9) \neq \mathrm{wd}'(v_9, v_3) = 11$ and $35 = \mathrm{wd}'(v_9, v_6) > (1+\varepsilon)\mathrm{wd}(v_9, v_6) = 7.7$, where the latter is feasible since $\mathrm{hd}(v_9, v_6) > 2$, i.e., $\mathrm{wd}_h(v_9, v_6) = \infty$

- $E_{\mathcal{S},h} = \{\{v, w\} \mid v, w \in \mathcal{S} \wedge v \neq w \wedge \mathrm{hd}(v, w) \leq h\}$;
- For $\{v, w\} \in E_{\mathcal{S},h}$, $W_{\mathcal{S},h}(v, w) = \mathrm{wd}_h(v, w)$.

We denote the distance function in $G_{\mathcal{S},h}$ by $\mathrm{wd}_{\mathcal{S},h}$.

It is straightforward to show (see Lemma 6.2) that if $\mathcal{S}$ is a uniformly random set of $c \cdot n \log n/h$ nodes, where $c$ is a sufficiently large constant, then with high probability, the distances in the skeleton graph are identical to the distances in $G$. In particular, choosing $h = \sqrt{n}$ allows us to preserve distances using a skeleton of size $|\mathcal{S}| \in \tilde{\Theta}(\sqrt{n})$.

An *$\alpha$-spanner*, for a given $\alpha \geq 1$, is a subgraph approximating distances up to factor $\alpha$. By computing a sufficiently sparse spanner of the skeleton graph—referred to as the *skeleton spanner*—we obtain a compact approximate representation of the skeleton graph which can be shipped to all nodes fast. To this end, we show how to simulate the $\mathcal{O}(k)$-round algorithm by Baswana and Sen [9] that, for an $n$-node graph, constructs (w.h.p.) a $(2k-1)$-spanner with $\tilde{\mathcal{O}}(n^{\frac{k+1}{k}})$ edges; for the skeleton graph this translates to $\tilde{\mathcal{O}}(n^{\frac{k+1}{2k}})$ edges. Each of the $k-1$ iterations of the algorithm is based on solving a (weighted) instance of $(\mathcal{S}, h, \sigma)$-detection, where $\sigma \in \tilde{\mathcal{O}}(n^{\frac{1}{k}})$, and can hence be completed in $\tilde{\mathcal{O}}(n^{\frac{k+1}{2k}})$ rounds. Pipelining the computed spanner to all nodes over a (single) BFS tree makes the skeleton spanner known to all nodes within a total number of $\tilde{\mathcal{O}}(n^{\frac{k+1}{2k}} + D)$ rounds.

**Theorem 1.7** *Let $\mathcal{S} \subseteq V$ be a random node set such that each $v \in V$ is in $\mathcal{S}$ independently with probability $c \log n/\sqrt{n}$, i.e., $\Pr[v \in \mathcal{S}] = \frac{c \log n}{\sqrt{n}}$, for a sufficiently large constant $c$. For any natural number $k \in \mathcal{O}(\log n)$, a $(2k-1)$-spanner of the $\lceil \sqrt{n} \rceil$-hop $\mathcal{S}$-skeleton graph can be computed and made known to all nodes in $\tilde{\mathcal{O}}(n^{\frac{k+1}{2k}} + D)$ rounds with high probability. Moreover, for each spanner edge $e = \{s, t\}$, there is a unique path $p_e$ from $s$ to $t$ in $G$ with the following properties:*

- *$p_e$ has weight $W_{\mathcal{S},\lceil\sqrt{n}\rceil}(e)$ (cf. Definition 1.6);*
- *$p_e$ has at most $\lceil \sqrt{n} \rceil$ hops;*
- *each node $v \in p_e \setminus \{s\}$ knows the next node $u \in p_e$ in the direction of $s$; and*
- *each node $v \in p_e \setminus \{t\}$ knows the next node $u \in p_e$ in the direction of $t$.*

The fact that the "magic number" $\sqrt{n}$ pops up repeatedly is no coincidence. As mentioned before, there is a well-known lower bound of $\tilde{\Omega}(\sqrt{n})$ that applies to a large class of problems in the **CONGEST** model, even when the hop diameter $D$ is very small [15,43]. Essentially, the issue is that while $D$ might be small, the shortest paths may induce a congestion bottleneck. We demonstrate that this is the case for APSP and routing table construction in Sect. 9.

### 1.1.4 Further results

The tools described above are applicable to many tasks. Below we give an informal overview of results that use them.

**Name-independent routing and distance approximation** In the routing table construction (RTC) problem, each node $v$ must compute a *routing table* such that given an identifier of a node $w$, $v$ can determine a neighbor $u$ based on its table and the identifier of $w$; querying $u$ for $w$ and repeating inductively, the route must eventually arrive at $w$. The *stretch* of the resulting path is its weight divided by the weight of a shortest $v-w$ path. The stretch of a routing scheme is the maximum stretch over all pairs of nodes.[3] In the distance approximation problem, the task is to output an approximate distance $\tilde{\mathrm{wd}}(v, w) \geq \mathrm{wd}(v, w)$ instead of the next routing hop when queried; the stretch then is the ratio $\tilde{\mathrm{wd}}(v, w)/\mathrm{wd}(v, w)$. Our algorithms always solve both RTC and distance approximation simultaneously, hence in what follows we drop the distinction and talk of "table construction".

The qualifier "name-independent", when applied to routing, refers to the fact that the algorithm is not permitted to assign new "names" to the nodes; as detailed below, such a reassignment may greatly reduce the complexity of the task. For name-independent table construction, the possible need to communicate $\Omega(n)$ identifiers over a bottleneck edge entails a running time lower bound of $\tilde{\Omega}(n)$, even in the unweighted case with $D \in \mathcal{O}(1)$. Close-to-optimal algorithms are given by solving source detection (in the

[3] Note that while this formulation of the routing problem does not deal directly with congestion as a cost measure, employing low-stretch routes reduces the network load and thus contributes towards a lower overall congestion. Also, sometimes edge weights represent the reciprocal of their bandwidth.

unweighted case, yielding stretch 1) or $(1+\varepsilon)$-approximate source detection (in the weighted case, yielding stretch $1+\varepsilon$) with $S = V$ and $\sigma = h = n$. (As an exception to the rule, these algorithms are deterministic; unless we indicate otherwise, in the following all results rely on randomization, and all lower bounds also apply to randomized algorithms.)

**Name-dependent routing and distance approximation** If table construction algorithms are permitted to assign to each node $v$ a (small) label $\lambda(v)$ and answer queries based on these labels instead, the game changes significantly. In this case, the strongest lower bounds are $\Omega(D)$ (trivial, also in unweighted graphs) and $\tilde{\Omega}(\sqrt{n})$; the latter applies even if $D \in \mathcal{O}(\log n)$. Combining approximate source detection and a skeleton spanner, we obtain tables of stretch $\mathcal{O}(k)$ in $\tilde{\mathcal{O}}(n^{\frac{k+2}{2k}} + D)$ rounds, with labels of optimal size $\mathcal{O}(\log n)$.

**Compact routing and distance approximation** In this problem, one adds the table size as an optimization criterion. It is straightforward to show that this implies that renaming must be permitted, as otherwise tables must comprise $\Omega(n \log n)$ bits, which is trivially achieved by evaluating the tables of any given scheme for all node identifiers (which can be made known to all nodes in $\mathcal{O}(n)$ rounds). We remark that one can circumvent this lower bound by permitting *stateful* routing, in which nodes may add auxilliaury bits to the message during the routing process. Intuitively, this makes it possible to distribute the large tables over multiple nodes, substantially reducing the degree of redundancy in stored information. In this article, we confine our attention to *stateless* routing, in which the routing decisions depend only on the destination's label and the local table.

Constructing a Thorup–Zwick routing hierarchy [52] by solving $k$ instances of source detection on unweighted graphs, we readily obtain tables of size $\tilde{\mathcal{O}}(n^{1/k})$ and stretch $\mathcal{O}(k)$ (this trade-off is known to be asymptotically optimal) within $\tilde{\mathcal{O}}(n^{1/k} + D)$ rounds. The weighted case is more involved: constructing the hierarchy through a skeleton spanner results in stretch $\Theta(k^2)$ for this table size and a target running time of $\tilde{\mathcal{O}}(n^{\frac{k+2}{2k}} + D)$ rounds. An alternative approach is to refrain from the use of a skeleton spanner and construct the hierarchy directly on the skeleton graph; this can be seen as constructing a spanner tailored to the routing scheme. Recently Elkin and Neiman (independently) pursued this direction, achieving stretch $4k - 5 + o(1)$ in $(n^{\frac{k+1}{2k}} + D)n^{o(1)}$ rounds [17].

**Single-source shortest paths and distance approximation** For single-source shortest paths (SSSP), the task is the same as in APSP, except that it suffices to determine routing information and distance estimates to a single node. Henziger et al. [24] employ approximate source detection to obtain a deterministic $(1 + o(1))$-approximation in near-optimal $n^{1/2+o(1)} + D^{1+o(1)}$ rounds. Their result is based on using approximate source detection to reduce the problem to an SSSP instance on an overlay network on $\tilde{\mathcal{O}}(\sqrt{n})$ nodes, which they then solve efficiently. The reduction itself does not incur an extra factor-$n^{o(1)}$ overhead in running time (beyond the $n^{1/2}$ factor). Indeed, very recent advances [10] result in a deterministic $(1 + o(1))$-approximation of the distances in $\tilde{\mathcal{O}}(\sqrt{n} + D)$ rounds, which is optimal up to a polylog $n$ factor. However, for extracting an approximate shortest path tree [10] relies on randomization. It is worth mentioning that the latter result makes use of a skeleton spanner to access a rough approximation of the distances in the skeleton, which it then "boosts" to a $(1 + o(1))$-approximation.

**Steiner forest** In the Steiner forest problem, we are given a weighted graph $G = (V, E, W)$ and disjoint terminal sets $V_1, \ldots, V_t$. The task is to find a minimum weight edge set $F \subseteq E$ so that for each $i \in \{1, \ldots, t\}$ and all $v, w \in V_i$, $F$ connects $v$ and $w$. Source detection and skeleton spanners have been leveraged in several distributed approximation algorithms for the problem [32–34].

**Tree embeddings** A tree embedding of a weighted graph $G = (V, E, W)$ maps its node set to the leaves of a tree $T = (V', E', W')$ so that $\mathrm{wd}_T(v, w) \geq \mathrm{wd}(v, w)$ (where $\mathrm{wd}_T$ denotes distances in the tree) and the expected stretch $\mathbb{E}[\mathrm{wd}_T(v, w)/\mathrm{wd}(v, w)]$ is small for each $v, w \in V$. Using a skeleton spanner, one can construct a tree embedding of expected stretch $\mathcal{O}(\varepsilon^{-1} \log n)$ in $\tilde{\mathcal{O}}(n^{1/2+\varepsilon} + D)$ rounds [22].

## 1.2 Organization of this paper

The remainder of the article is organized in a modular way. In the next section, we discuss related work. In Sect. 3, we specify the notation used throughout this paper and give formal definitions of the routing table construction problem and its variants; readers who already feel comfortable with the terms that appeared up to this point are encouraged to skip this section and treat it as a reference to be used when needed. We then follow through with fairly self-contained sections proving our claims: source detection (Sect. 4), approximate source detection (Sect. 5), skeleton and skeleton spanners (Sect. 6), table construction in unweighted graphs (Sect. 7), table construction in weighted graphs (Sect. 8), and lower bounds (Sect. 9).

## 2 Related work

### 2.1 Distributed algorithms for exact all-pairs shortest-paths

The exact all-pairs shortest path (APSP) problem has been studied extensively in the sequential setting, and was also given several solutions in the distributed setting [2,13,23,29,51]. The algorithm by Kanchi and Vineyard [29] is fast (runs in $\mathcal{O}(n)$ time) but involves using large messages, hence does not apply in the **CONGEST** model. The algorithm of Antonio et al. [2] uses short (i.e., $\mathcal{O}(\log n)$ bits) messages, hence it can be executed in the **CONGEST** model, but it requires $\mathcal{O}(n \log n)$ time, and moreover, it applies only to the special family of BHC graphs, which are graphs structured as a balanced hierarchy of clusters. Most of the distributed algorithms for the APSP problem aim at minimizing the message complexity, rather than the time; for instance, the algorithm of Haldar [23] requires $\mathcal{O}(n^2)$ time. For unweighted networks, a trivial lower bound of $\Omega(n)$ applies for exact APSP in the **CONGEST** model, as $\Omega(n)$ node identifiers may have to be communicated through a bottleneck edge. This lower bound has been matched (asymptotically) by two distributed $\mathcal{O}(n)$-time algorithms, proposed independently by Holzer and Wattenhofer [26] and Peleg et al. [42]. Apart from solving a more general problem, our solution slightly improves on each of these algorithms. Compared to the first algorithm, our solution attains the optimal time with respect to the constant factors (cf. Corollary 7.1). Compared to the second, our algorithm never sends different messages to different neighbors in the same round.

For weighted networks, prior to this work there has been little progress from the theoretical perspective on computing weighted shortest paths faster than the SPD barrier, where SPD ("shortest paths diameter") is minimal with the property that $\mathrm{wd}_{\mathrm{SPD}} = \mathrm{wd}$, i.e., between each pair of nodes there is a shortest path of at most SPD hops; see, e.g., Das Sarma et al. [14] and references therein.

#### 2.1.1 Distributed construction of compact routing tables

There are many centralized algorithms for constructing compact routing tables (a routing table at a node says which hop to take for each possible destination); in these algorithms the goal is usually to minimize space without affecting the quality of the routes too badly. Following the early constructions in [3,6,46], Thorup and Zwick [52] presented an algorithm that achieves, for any $k \in \mathbb{N}$, routes of stretch at most $2k-1$ using $\tilde{\mathcal{O}}(n^{1/k})$ memory, which is optimal up to a constant factor in worst-case stretch w.r.t. routing [46]. Note that a naïve distributed implementation of a centralized algorithm in the **CONGEST** model requires $\Omega(|E|)$ time in the worst case, since the whole network topology has to be collected at a single node.

Practical distributed routing table construction algorithms are usually categorized as either "distance vector" or "link state" algorithm (see, e.g., Peterson and Davie [47]). Distance-vector algorithms are variants of the Bellman–Ford algorithm [11,18], whose worst-case time complexity in the **CONGEST** model is $\Theta(n^2)$. In link-state algorithms [37,38], each routing node collects the complete graph topology and then solves the single-source shortest path problem locally. This approach has $\Theta(|E|)$ time complexity.

Both the approximate shortest paths and the compact routing problems have been studied extensively. However, most previous work on these problems either focused on efficient performance (stretch, memory) and ignored the preprocessing stage (cf. [6,20,41,46] and references), or provided time-efficient *sequential (centralized)* preprocessing algorithms [7,8,30,49,49,53]. Relatively little attention was given to *distributed* preprocessing algorithms, and previous work on such algorithms either ignored time-efficiency (cf. [3,4]) or assumed a model allowing large messages (cf. [5]).

#### 2.1.2 Spanners

A closely related concept is that of *sparse graph spanners* [44,45]. It is known that a $(2k-1)$-spanner must have $\tilde{\Omega}(n^{1+1/k})$ edges for some values of $k$, and this lower bound is conjectured to hold for all $k \in \mathbb{N}$. A matching upper bound is obtained by the construction of Thorup and Zwick [53]. Our construction of skeleton spanners simulates an elegant algorithm by Baswana and Sen [9] on the skeleton. Their algorithm achieves stretch at most $2k-1$ vs. $\mathcal{O}(n^{1+1/k})$ expected edges within $\mathcal{O}(k)$ rounds in the **CONGEST** model. A deterministic construction with similar performance but allowing large messages is presented by Derbel et al. [16].

#### 2.1.3 Distributed lower bounds

Lower bounds of $\tilde{\Omega}(\sqrt{n})$ on the running time of a variety of global distributed problems (including MST, shortest-paths tree of low weight, and stateless routing) were presented by Das Sarma et al. [15] and Peleg and Rubinovich [43]. Without relabeling (i.e., when renaming of the nodes is forbidden), routing table construction and APSP both require $\Omega(n)$ rounds [32,39], regardless of the stretch or approximation ratio, respectively. Another (almost) linear-time barrier arises from the approximation ratio: any approximation of the hop diameter better than factor $3/2$ [1,19] or the weighted diameter (the maximum weight of a shortest path) better than 2 [25] takes $\Omega(n/\log n)$ rounds. A matching upper bound of $\mathcal{O}(n/\log n + D)$ rounds for exact APSP in unweighted graphs (with relabeling) proves this bound to be asymptotically tight

[27], as it immediately implies that the hop diameter can be computed in the same time. Izumi and Wattenhofer [28] prove a lower bound of $\tilde{\Omega}(n^{1/(t+1)})$ (and a lower bound of $\tilde{\Omega}(n^{\frac{1}{2}+\frac{1}{5t}})$) on the running time required to compute in the **CONGEST** model a labeling scheme that allows one to estimate the distances with stretch at most $2t$ in unweighted graphs (and in weighted graphs, respectively).

#### 2.1.4 Leveraging the shortest-path-diameter

Das Sarma et al. [14] show how to construct distance tables of size $\tilde{\mathcal{O}}(n^{1/k})$ with stretch $2k-1$ in the **CONGEST** model in $\tilde{\mathcal{O}}(\mathrm{SPD}\, n^{1/k})$ rounds, where SPD is the minimal hop count such that between any two nodes there is a weighted shortest path of at most SPD hops. They exploit the Bellman–Ford relaxation with termination detection via an (unweighted) BFS tree within $\mathcal{O}(D)$ time. Our analysis enables us to generalize this result using small labels, albeit with stretch $4k-3$ (Corollary 8.1); this is because, unlike in [14], we disallow access to the destination's table.

## 3 Preliminaries

In this section we define the model of computation, introduce some notation, and discuss a few basic subroutines we make explicit or implicit use of frequently.

### 3.1 The computational model

We follow the **CONGEST** model as described in [41]. The distributed system is represented by a simple, connected weighted graph $G = (V, E, W)$, where $V$ is the set of nodes, $E$ is the set of edges, and $W : E \to \mathbb{N}$ is the edge weight function.[4] As a convention, we use $n$ to denote the number of nodes, and assume that all edge weights are bounded by some polynomial in $n$, and that each node $v \in V$ has a unique identifier of $\mathcal{O}(\log n)$ bits, to conform with the **CONGEST** model [41]. (We use $v$ to denote both the node and its identifier.)

Execution proceeds in global synchronous rounds, where in each round, each node takes the following three steps:

(1) Perform local computation,
(2) send messages to neighbors, and
(3) receive the messages sent by neighbors.

Moreover, a node may decide to terminate and output a result at the end of any given round. A node that terminated ceases to execute the above steps. The running time or *round complexity* of a deterministic algorithm is the worst-case number of rounds (parametrized with $n$, $D$, etc.) until all nodes have terminated. For randomized algorithms, the respective bound may hold with a certain probability bound only.

Initially, nodes have the following information:

- their own identifier;
- the identifiers of the respective other endpoint of incident edges;[5]
- the weight of incident edges (if the graph is weighted); and, in general,
- possible further problem-specific input.

In each round, each edge can carry a message of $B$ bits for some given parameter $B$ of the model. Throughout this article, we make the common assumption that $B \in \Theta(\log n)$.

### 3.2 General concepts

We use $\mathbb{N}$ to denote the natural numbers, $\mathbb{N}_0$ to denote $\mathbb{N} \cup \{0\}$, and $\mathbb{N}_\infty$ to denote $\mathbb{N} \cup \{\infty\}$.

Given a list $L = \langle a_1, a_2, \ldots, a_\ell \rangle$ and $k \in \mathbb{N}$, we use $\mathrm{top}_k(L)$ to denote the list that consists of the first $k$ elements of $L$, or $L$ if $\ell < k$.

We use extensively "soft" asymptotic notation that ignores polylogarithmic factors. Formally, $f(n) \in \tilde{\mathcal{O}}(g(n))$ if and only if there exists a constant $c \in \mathbb{R}_0^+$ such that $f(n) \leq g(n)\log^c(n)$ for all but finitely many values of $n \in \mathbb{N}$. Analogously,

- $f(n) \in \tilde{\Omega}(g(n))$ iff $g(n) \in \tilde{\mathcal{O}}(f(n))$,
- $\tilde{\Theta}(f(n)) = \tilde{\mathcal{O}}(f(n)) \cap \tilde{\Omega}(f(n))$,
- $f(n) \in \tilde{o}(g(n))$ iff for any $c \in \mathbb{R}_0^+$ it holds that $\limsup_{n\to\infty} f(n)\log^c(n)/g(n) = 0$, and
- $f(n) \in \tilde{\omega}(g(n))$ iff $g(n) \in \tilde{o}(f(n))$.

Note that polylog $n = \tilde{\mathcal{O}}(1)$.

To model probabilistic computation, we assume that each node has access to an infinite string of independent unbiased random bits. When we say that a certain event occurs "with high probability" (abbreviated "w.h.p."), we mean that the probability of the event not occurring can be set to be less than $1/n^c$ for any desired constant $c$, where the probability is taken over the strings of random bits. As $c$ is meant to be a constant, it will be hidden by asymptotic notation. We remark that for all our results, $c$ affects the time complexity at most as a multiplicative factor.

[4] We remark that our results can be easily extended to non-negative edge weights by employing appropriate symmetry breaking mechanisms.

[5] This assumption is made for notational convenience; it takes a single round to exchange identifiers with neighbors.

### 3.3 Some graph-theoretic concepts

We consider both weighted and unweighted graphs; in weighted graphs, we use $W : V \to \mathbb{N}$ to denote the weight function, and assume that edge weights are bounded by $n^{\mathcal{O}(1)}$. With the exception of Sects. 4.4 and 5.3, we consider undirected graphs and assume this to be the case without further notice. Without loss of generality, graphs are simple; self-loops as well as all but a lightest edge between a pair of nodes can be deleted without changing the solutions, and thus worst-case instances will not provide additional communication bandwidth due to parallel edges.

A *path* $p$ connecting $v, u \in V$ is a finite sequence of nodes $\langle v = v_0, \dots, v_k = u \rangle$ such that for all $0 \leq i < k$, $\{v_i, v_{i+1}\}$ is an edge in $G$. Let $\text{paths}(v, u)$ denote the set of all paths connecting nodes $v$ and $u$. (This set may contain also non-simple paths, but our focus later on is on *shortest* paths, which are always simple.) We use the following unweighted concepts.

- The *hop-length* of a path $p$, denoted $\ell(p)$, is the number of edges in it.
- A path $p_0$ between $v$ and $u$ is a *shortest unweighted path* if its hop-length $\ell(p_0)$ is minimum among all $p \in \text{paths}(v, u)$.
- The *hop distance* $\text{hd} : V \times V \to \mathbb{N}_0$ is defined as the hop-length of a shortest unweighted path, $\text{hd}(v, u) := \min\{\ell(p) \mid p \in \text{paths}(v, u)\}$.
- The *(hop-)diameter* $D = \max_{v,u \in V}\{\text{hd}(v, u)\}$.

We use the following weighted concepts.

- The *weight* of a path $p$, denoted $W(p)$, is its total edge weight, i.e., $W(p) = \sum_{i=1}^{\ell(p)} W(v_{i-1}, v_i)$.
- A path $p_0$ between $v$ and $u$ is a *shortest weighted path* if its weight $W(p_0)$ is minimum among all $p \in \text{paths}(v, u)$.
- The *weighted distance* $\text{wd} : V \times V \to \mathbb{N}$ is defined as the weight of a shortest weighted path,

  $$\text{wd}(v, u) = \min\{W(p) \mid p \in \text{paths}(v, u)\}.$$

- The *weighted diameter*

  $$\text{WD} = \max\{\text{wd}(v, u) \mid v, u \in V\}.$$

Finally, we define the following "hybrid" notions.

- For $h \in \mathbb{N}$,

  $$\text{wd}_h(v, u) = \inf\{W(p) \mid p \in \text{paths}(v, u) \wedge \ell(p) \leq h\}$$

  is the *h-hop distance*. Note that $\text{wd}_h(v, u) = \infty$ iff $\text{hd}(v, u) > h$.
- The *shortest path diameter* is

  $$\text{SPD} = \min\{h \in \mathbb{N} \mid \text{wd}_h = \text{wd}\},$$

  i.e., the minimum hop distance $h$ so that for each $u, v \in V$ there is a shortest weighted path of at most $h$ hops.

Note that for $h < \text{SPD}$, $\text{wd}_h$ is not a metric, as it violates the triangle inequality.

### 3.4 Basic primitives

The results in this section can be considered folklore. We will informally sketch the basic algorithmic ideas. For a more detailed exposition, we refer to [41].

Based on a simple flooding, it is straightforward to construct a BFS tree rooted at any given node in $D$ rounds. By starting this routine concurrently for each node as a potential root, but ignoring all instances except for the one corresponding to the node of (so far) smallest known identifier, one constructs a single BFS tree and implicitly elects a leader. By reporting back to the root via the (so far) constructed tree whenever a new node is added, the root detects that the tree was completed by round $2D + 2$.

**Lemma 3.1** *A single BFS tree can be constructed in $\Theta(D)$ rounds. Moreover, the root learns the depth $d \in [D/2, D]$ of the tree.*

Most problems discussed in this article are *global*, i.e., satisfy trivial running time lower bounds of $\Omega(D)$. By the above lemma, we can hence assume that termination is coordinated by the root of a BFS tree without affecting asymptotic running times: nodes report to their parent when the subtree rooted at them is ready to terminate, and once the root learns that all nodes are ready, it can decide that all nodes shall terminate $d$ rounds later and distribute this information via the tree. Accordingly, we will in most cases refrain from discussing how nodes decide on when to terminate.

A BFS tree supports efficient basic operations, such as broadcasts and convergecasts. In particular, it can be used to determine sums, maxima, or minima of individual values held by the nodes.

**Lemma 3.2** *Within $\Theta(D)$ rounds, the following can be determined and made known to all nodes:*

- *The number of nodes $n$.*
- *The maximum edge weight* $\max_{e \in E}\{W(e)\}$.
- *The minimum edge weight* $\min_{e \in E}\{W(e)\}$.
- *$|S|$ for any $S \subseteq V$ given locally, i.e., when each $v \in V$ knows whether $v \in S$ or not.*

Therefore, we may assume w.l.o.g. that such values are globally known in our algorithms. For simplicity, we will

also assume that $D$ is known; in practice, one must of course rely on the upper bound $2d \in [D, 2D]$ instead, at the expense of a constant-factor increase in running times.

In addition, we will make excessive use of *pipelining*, i.e.,running multiple broadcast and convergecast operations on the BFS tree concurrently.

**Lemma 3.3** *Suppose each $v \in V$ holds $m_v \in \mathbb{N}_0$ messages of $\mathcal{O}(\log n)$ bits each, for a total of $M = \sum_{v \in V} m_v$ strings. Then all nodes in the graph can receive these $M$ messages within $\mathcal{O}(M + D)$ rounds.*

In the following, we use this lemma implicitly whenever stating that some information is "broadcast to all nodes" or "announced to all nodes".

## 4 Source detection in unweighted graphs

In this section, we present an efficient deterministic algorithm for the source detection task on unweighted graphs. Accordingly, we assume that the graph is unweighted throughout this section. Recall the task we need to solve:

**Definition 4.1** (*Unweighted $(S, h, \sigma)$-detection, restated*) Given $S \subseteq V$, a node $v \in V$, and non-negative integer $h \in \mathbb{N}_0$, let $\mathcal{L}_v^{(h)}$ be the list of elements $\{(\mathrm{hd}(v, s), s) \mid s \in S \wedge \mathrm{hd}(v, s) \leq h\}$, ordered in ascending lexicographical order. For $\sigma \in \mathbb{N}$, $(S, h, \sigma)$-*detection* requires each node $v \in V$ to compute $\mathrm{top}_\sigma(\mathcal{L}_v^{(h)})$.

Without restrictions on bandwidth, a variant of the Bellman–Ford algorithm solves the problem in $\mathcal{O}(h)$ time. Each node $v$ maintains a list $L_v$ of the (distance, source) pairs that it knows about. $L_v = \emptyset$ if $v \notin S$, and $L_v = \{(0, v)\}$ if $v \in S$. In each round, each node $v$ sends $L_v$ to its neighbors. Upon reception of such a message, for each received pair $(h, s)$ for which there is no own pair $(h', s) \in L_v$, it adds $(h + 1, s)$ to $L_v$. After $h$ rounds, $v$ knows the sources within hop distance $h$ from itself and their correct hop distance; thus it is able to order the source/distance pairs correctly. This approach concurrently constructs BFS trees up to depth $h$ for all sources $s \in S$.

### 4.1 Pipelined Bellman–Ford algorithm

A naïve implementation of the above algorithm in the **CONGEST** model would cost $\mathcal{O}(\sigma h)$ time, since messages contain up to $\sigma$ pairs, each of $\mathcal{O}(\log n)$ bits. However, it turns out that in the unweighted case, the following simple idea works: in each round, each node $v \in V$ announces only the smallest pair $(h, s)$ in $L_v$ it has not announced yet. Pseudocode is given in Algorithm 1. (The algorithm can be trivially extended to construct BFS trees rooted at the sources.)

**Algorithm 1:** PBF$(S, h, \sigma)$: Pipelined Bellman–Ford at node $v \in V$.

```
   input : S: sources                    // v ∈ V knows if v ∈ S
           h: distance parameter
           σ: number of sources to detect
   output: list L_v
   // list of distance/source pairs
       (h_s, s) ∈ ℕ_0 × S
1  L_v := ∅
   // whether a pair in L_v has been sent yet
2  sent_v : L_v → {TRUE, FALSE}
3  if v ∈ S then
4      L_v := {(0, v)}
5      sent_v(0, v) := FALSE
   // one round per iteration
6  for h + σ − 1 iterations do
7      if ∃(h_s, s) ∈ L_v : sent_v(h_s, s) = FALSE then
8          (h_s, s) := argmin{(h_s', s') ∈ L_v | sent_v(h_s', s') = FALSE}
9          send (h_s, s) to all neighbors
10         sent_v(h_s, s) := TRUE
11     for (h_s, s) received from some neighbor do
12         if ∄(h_s', s) ∈ L_v : h_s' ≤ h_s + 1 then
               // remove outdated entry (if exists)
13             L_v := L_v \ {(·, s)}
14             L_v := L_v ∪ {(h_s + 1, s)}
15             sent_v(h_s + 1, s) := FALSE
16 delete all entries (h_s, s) from L_v with h_s > h
17 return top_σ(L_v)
```

### 4.2 Analysis

The algorithm appears simple enough, but note that since only one pair is announced by each node in every round, it may now happen that a pair $(h, s)$ is stored in $L_v$ with $h > \mathrm{hd}(v, s)$. Further, we need to consider that $v$ might announce this pair to other nodes. However, nodes keep announcing smaller distances as they learn about them, and eventually $L_v = \{(\mathrm{hd}(s, v), s) \mid s \in S\}$ for all $v \in V$.

To prove this formally, we first fix some helpful notation.

**Definition 4.2** For each node $v \in V$ and each round $r \in \mathbb{N}$, denote by $L_v^r$ the content of $v$'s $L_v$ variable at the end of round $r$; by $L_v^0$ we denote the value at initialization.

We start with the basic observation that, at all times, list entries may be incorrect only in that the stated distances may be too large.

**Lemma 4.3** *For all $v \in V$ and $r \in \mathbb{N}_0$: If $(h_s, s) \in L_v^r$, then $s \in S$ and $h_s \geq \mathrm{hd}(v, s)$.*

***Proof*** By induction on $r$. For $r = 0$ the claim holds by Lines 3–4. For the inductive step, assume that the claim holds for $r \in \mathbb{N}_0$ and consider $r + 1$. If $(h_s, s) \in L_v^r$ we are done by the induction hypothesis. Thus, consider a message $(h, s)$ received at time $r + 1$. First note that by Line 9, which is the only place where messages are sent, $(s, h) \in L_u^r$ for some

neighbor $u$ of $v$. Hence, by the induction hypothesis applied to $u$, $s \in S$. Now suppose that $(h+1, s)$ is inserted into $L_v$ in Line 14. By the induction hypothesis, we have that $h \geq \mathrm{hd}(u, s)$, and hence, using the triangle inequality, we may conclude that $h + 1 \geq \mathrm{hd}(u, s) + 1 \geq \mathrm{hd}(v, s)$, as required. □

This immediately implies that (i) correct pairs will never be deleted and (ii) if a prefix of $\mathcal{L}_v^{(h)}$ is known to $v$, $v$ will communicate this prefix to all neighbors before sending "useless" pairs.

**Corollary 4.4** *Let $s \in S$ and $v \in V$. If $v$ receives $(\mathrm{hd}(v, s) - 1, s)$ from a neighbor in round $r \in \mathbb{N}$, or if $(\mathrm{hd}(v, s), s) \in L_v^0$, then $(\mathrm{hd}(v, s), s) \in L_v^{r'}$ for all $r' \geq r$. Moreover, if $\mathrm{top}_k(\mathcal{L}_v^{(h)}) \subseteq L_v^r$ for any $r \in \mathbb{N}_0$ and $k \in \mathbb{N}$, then $\mathrm{top}_k(L_v^r) = \mathrm{top}_k(\mathcal{L}_v^{(h)})$.*

In particular, it suffices to show that $\mathrm{top}_\sigma(\mathcal{L}_v^{(h)}) \subseteq L_v^r$ at termination. Before we move on to the main lemma, we need another basic property that goes almost without saying: a source $s \in S \setminus \{v\}$ that is among the $k$ closest sources to $v$ must also be among the $k$ closest sources of a neighbor $w$ with $\mathrm{hd}(w, s) = \mathrm{hd}(v, s) - 1$.

**Lemma 4.5** *For all $h, k \in \mathbb{N}$ and all $v \in V$,*

$$\mathrm{top}_k(\mathcal{L}_v^{(h)}) \subseteq \mathcal{L}_v^{(0)} \cup \Big\{(\mathrm{hd}(w, s) + 1, s) \mid (\mathrm{hd}(w, s), s) \in \mathrm{top}_k(\mathcal{L}_w^{(h-1)}) \wedge \{v, w\} \in E\Big\}.$$

***Proof*** For any $(\mathrm{hd}(v, s), s) \in \mathrm{top}_k(\mathcal{L}_v^{(h)}) \setminus \mathcal{L}_v^{(0)}$, consider a neighbor $w$ of $v$ on a shortest path from $v$ to $s$. We have that $\mathrm{hd}(w, s) = \mathrm{hd}(v, s) - 1$, i.e., $(\mathrm{hd}(w, s), s) \in \mathcal{L}_w^{(h-1)}$. Assume for contradiction that $(\mathrm{hd}(w, s), s) \notin \mathrm{top}_k(\mathcal{L}_w^{(h-1)})$. Then there are $k$ elements $(\mathrm{hd}(w, s'), s') \in \mathrm{top}_k(\mathcal{L}_w^{(h-1)})$ satisfying $(\mathrm{hd}(w, s'), s') < (\mathrm{hd}(w, s), s)$. Hence, for each of these elements, $(\mathrm{hd}(v, s'), s') \leq (\mathrm{hd}(w, s') + 1, s') < (\mathrm{hd}(w, s) + 1, s) = (\mathrm{hd}(v, s), s)$, and hence $(\mathrm{hd}(v, s), s) \notin \mathrm{top}_k(\mathcal{L}_v^{(h)})$, a contradiction. □

We are now ready to prove the key invariants of the algorithm.

**Lemma 4.6** *Let $v \in V$, $r \in \{0, \ldots, h+\sigma-1\}$, and let $d, k \in \mathbb{N}_0$ be such that $d + k \leq r + 1$. Then (i) $\mathrm{top}_k(\mathcal{L}_v^{(d)}) \subseteq L_v^r$; and (ii) by the end of round $r + 1$, if not terminated, $v$ sends $\mathrm{top}_k(\mathcal{L}_v^{(d)})$.*

***Proof*** By induction on $r$. The statement trivially holds for $d = 0$ and all $k$, as $\mathrm{top}_k(\mathcal{L}_v^{(d)}) = \{(0, v)\}$ if $v \in S$ and $\mathrm{top}_k(\mathcal{L}_v^{(d)}) = \emptyset$ otherwise, and clearly this will be sent by the end of round 1. In particular, the claim holds for $r = 0$.

Now suppose that the statement holds for $r$ and consider $r + 1$. To this end, fix some $d + k \leq r + 2$, where we may assume that $d > 0$, because the case $d = 0$ is already covered.

By part (ii) of the induction hypothesis applied to $r$ for values $d - 1$ and $k$, node $v$ has already received the lists $\mathrm{top}_k(\mathcal{L}_w^{(d-1)})$ from all neighbors $w$. By Lemma 4.5, $v$ thus has received all elements of $\mathrm{top}_k(\mathcal{L}_v^{(d)})$. By Corollary 4.4, this implies Statement (i) for $d + k \leq r + 2$.

It remains to show (ii) for $d + k = r + 2 \leq h + \sigma - 1$. Since we just have shown (i) for $d + k = r + 2$, we know that $\mathrm{top}_k(\mathcal{L}_v^{(d)}) \subseteq L_v^{r+1}$ for all $d, k$ satisfying $d + k = r + 2$. By Corollary 4.4, these are actually the first elements of $L_v^{r+1}$, so $v$ will send the next unsent entry of $\mathrm{top}_k(\mathcal{L}_v^{(d)})$ in round $r + 2$ (if there is one). As $d + (k - 1) = r + 1$, we can apply the induction hypothesis to see that $v$ sent $\mathrm{top}_{k-1}(\mathcal{L}_v^{(d)})$ during the first $r + 1$ rounds (where we define $\mathrm{top}_0(\mathcal{L}_v^{(d)}) = \emptyset$). Hence, only $\mathrm{top}_k(\mathcal{L}_v^{(d)}) \setminus \mathrm{top}_{k-1}(\mathcal{L}_v^{(d)})$ may still be missing. As $\left|\mathrm{top}_k(\mathcal{L}_v^{(d)}) \setminus \mathrm{top}_{k-1}(\mathcal{L}_v^{(d)})\right| \leq 1$ by definition, this proves (ii) for $d + k = r + 2$. This completes the induction step and thus the proof. □

The reader may wonder why the final argument in the above proof addresses all possible combinations of $d + k = r + 2$ simultaneously. This is true because the missing element (if any) is the same for all such values. To see this, observe the following: (i) if $|\mathrm{top}_k(\mathcal{L}_v^{(d)})| < k$, then $\mathrm{top}_{k-1}(\mathcal{L}_v^{(d)}) = \mathrm{top}_k(\mathcal{L}_v^{(d)})$ and no entry needs to be sent; (ii) if $|\mathrm{top}_k(\mathcal{L}_v^{(d)})| = k$, then $\mathrm{top}_k(\mathcal{L}_v^{(d')}) = \mathrm{top}_k(\mathcal{L}_v^{(d)}) \supseteq \mathrm{top}_{k-(d'-d)}(\mathcal{L}_v^{(d)})$ for all $d' \geq d$. Accordingly, for all $d$ and $k$ for which still an entry needs to be sent, it is the same.

We are now ready to prove our first main result, Theorem 1.2, showing that unweighted $(S, h, \sigma)$-detection can be solved in $\sigma + h - 1$ rounds.

***Proof of Theorem 1.2.*** By Lemma 4.6, $\mathrm{top}_\sigma(\mathcal{L}_v^{(h)}) \subseteq L_v^{h+\sigma-1}$. By Corollary 4.4, $\mathrm{top}_\sigma(\mathcal{L}_v^{(h)}) = \mathrm{top}_\sigma(L_v^{h+\sigma-1})$, implying that Algorithm 1 returns the correct output, which establishes the theorem. □

We remark that one can generalize this result to show that if up to $\beta$ list entries are sent in a message, $(S, h, \sigma)$-detection is solved within $h + \lceil \sigma/\beta \rceil - 1$ rounds. Likewise, we have a trivial lower bound of $h + \lceil \sigma/\beta \rceil - 1$ for $(S, h, \sigma)$-detection in this setting. Our technique is thus essentially optimal.

### 4.3 Additional properties

We conclude this section with a few observations that we use later. First, if we use Algorithm 1 to construct partial BFS trees of depth $h$ rooted at the sources $S$ (i.e., $\sigma = |S|$), we get a schedule that facilitates flooding or echo on all partial BFS trees concurrently in $\sigma + h$ rounds.

**Corollary 4.7** *Consider an execution of Algorithm 1. Let $p_s(v)$ denote the node from which a node $v$ receives the message $(\mathrm{hd}(s,v)-1,s)$ for the first time, where $(\mathrm{hd}(s,v),s)$ is in the output $L_v$ of $v$. Then*

(i) *All these messages are received by round $h+\sigma$ of the execution.*
(ii) *The edges $\{(v,p_s(v)) \mid v \in V \setminus \{s\}\}$ induce a BFS tree rooted at $s$, comprising only nodes within distance at most $h$ from $s$. (If $\sigma \geq |S|$, the tree comprises all such nodes.)*
(iii) *The sending pattern of these messages defines a schedule for* concurrent flooding *on all such trees. In the concurrent flooding operation, each node in a tree sends a message of its choice to all neighbors (in particular its children in the tree), such that on any root-leaf path the sending order matches the order of nodes in the path. Thus, each inner node is scheduled before any of its children, and its message may depend on the messages sent by its parent.*
(iv) *If the sending pattern of these messages is reversed (after running the algorithm for one more round and removing the first round of the execution), this defines a schedule for* concurrent echo *on all such trees. In the concurrent echo operation, each node in a tree sends a message of its choice to all neighbors (in particular its parent in the tree) such that on any leaf-root path the sending order matches the order of nodes in the path. Thus, each inner node receives the messages of all its children before sending its own, i.e., its message may depend on those of its children.*

***Proof*** The first statement follows directly from Theorem 1.2. The second follows by observing that for each $v$ with $\mathrm{hd}(s,v) \leq h$, by construction $p_s(v)$ is in hop distance $\mathrm{hd}(s,v)-1$ from $s$. The third statement holds because $v$ cannot send the message $(\mathrm{hd}(s,v),s)$ before receiving $(\mathrm{hd}(s,v)-1,s)$ for the first time. The last statement immediately follows from the third (for $h+1$). □

Note that, in particular, storing the parent relation for the BFS trees is sufficient for routing purposes.

**Corollary 4.8** *Algorithm 1 can be used to construct routing tables of $\mathcal{O}(\sigma \log n)$ bits for destinations in $L_v$.*

### 4.4 Source detection in unweighted directed graphs

While this article studies distance problems in undirected graphs, it is worth mentioning that our source detection primitives work equally well on directed graphs. Note that in a directed graph, $\mathrm{hd}(v,w) \neq \mathrm{hd}(w,v)$, where $\mathrm{hd}(v,w)$ is the minimum hop count of a directed path from $v$ to $w$.

**Definition 4.9** (*Unweighted Directed $(S,h,\sigma)$-detection*) Given an unweighted directed graph $G=(V,E)$, define for $v,w \in V$ that $\mathrm{hd}(v,w)$ is the minimum number of hops on a directed path from $v$ to $w$.

Given $S \subseteq V$, a node $v \in V$, and non-negative integer $h \in \mathbb{N}_0$, let $\mathcal{L}_v^{(h)}$ be the list of elements $\{(\mathrm{hd}(s,v),s) \mid s \in S \wedge \mathrm{hd}(s,v) \leq h\}$, ordered in ascending lexicographical order. For $\sigma \in \mathbb{N}$, *directed $(S,h,\sigma)$-detection* requires each node $v \in V$ to compute $\mathrm{top}_\sigma(\mathcal{L}_v^{(h)})$.

It is straightforward to verify that the reasoning from this section applies analogously to the directed case.

**Corollary 4.10** *If we execute Algorithm 1 on a directed graph such that messages are sent only to out-neighbors, this solves the unweighted directed $(S,h,\sigma)$-detection problem in $\sigma+h-1$ rounds.*

We note that this corollary applies even if communication is only possible in direction of the graph edges. However, for performing echo operations as per Corollary 4.7, detecting termination using a BFS tree, or determinining and making known parameters like the number of nodes, bidirectional communication is necessary.

## 5 Approximate source detection

We now consider source detection in weighted graphs, approximately. We recall the definition.

**Definition 5.1** (*Approximate Source Detection, restated*) Given $S \subseteq V$, $h,\sigma \in \mathbb{N}$, and $\varepsilon > 0$, let $\mathcal{L}_v^{(h,\varepsilon)}$ be a list of $\{(\mathrm{wd}'(v,s),s) \mid s \in S,\ \mathrm{wd}'(v,s) < \infty\}$, ordered in increasing lexicographical order, for some $\mathrm{wd}' : V \times S \to \mathbb{N}_\infty$ that satisfies $\mathrm{wd}'(v,s) \in \left[\mathrm{wd}(v,s),\ (1+\varepsilon)\mathrm{wd}_h(v,s)\right]$ for all $v \in V$ and $s \in S$. The $(1+\varepsilon)$-*approximate* $(S,h,\sigma)$-*detection* problem is to output $\mathrm{top}_\sigma(\mathcal{L}_v^{(h,\varepsilon)})$ at each node $v$ for some such $\mathrm{wd}'$.

### 5.1 Reduction to the unweighted case

Fix $0 < \varepsilon \leq 1$ and natural $h < n$. Following Nanongkai [39] and others [12,31,36,48], we reduce approximate weighted source detection to $\mathcal{O}(\log_{1+\varepsilon} n)$ instances of the exact unweighted problem. The main idea is to round edge weights to integer multiples of $(1+\varepsilon)^i$ and replace each edge with a path consisting of the respective number of unit weight edges. One then shows that for each shortest path, there is a "good" choice of $i \in \mathcal{O}(\log_{1+\varepsilon} n)$ such that its weight is approximately preserved, yet its hop count does not increase too much.

Formalizing this approach, we define $i_{\max} = \lceil \log_{1+\varepsilon}(h \max_{e \in E}\{W(e)\}) \rceil$, i.e., $i_{\max}$ is the logarithm, to base $(1+\varepsilon)$, of (an upper bound on) the maximum weight of paths of

$h$ hops. Note that by our assumption on the magnitude of weights, $i_{\max} \in \mathcal{O}(\log_{1+\varepsilon} n)$.

For $i \in \{0, \ldots, i_{\max}\}$, define $b(i) = (1+\varepsilon)^i$, and

$$\forall e \in E : \; W_i(e) = b(i)\lceil W(e)/b(i) \rceil ,$$

i.e., $W_i(e)$ is $W(e)$ rounded up to the next integer multiple of $(1+\varepsilon)^i$. Let $\mathrm{wd}_i$ denote the distance function of the graph $(V, E, W_i)$. Then the following crucial property holds.

**Lemma 5.2** (adapted from [39]) *Given $0 < \varepsilon \leq 1$ and distinct nodes $v, w \in V$ with $\mathrm{hd}(v, w) \leq h$, let*

$$i_{v,w} = \max\left\{0, \left\lfloor \log_{1+\varepsilon}\left(\frac{\varepsilon\, \mathrm{wd}_h(v,w)}{h}\right)\right\rfloor\right\} .$$

*Then*

$$\mathrm{wd}_{i_{v,w}}(v,w) < (1+\varepsilon)\mathrm{wd}_h(v,w) < \frac{4b(i_{v,w})h}{\varepsilon} .$$

***Proof*** For $i_{v,w} = 0$ we have $\mathrm{wd}_0 = \mathrm{wd}$ because $b(0) = 1$ and clearly $\mathrm{wd}(v,w) < (1+\varepsilon)\mathrm{wd}_h(v,w)$ for $\varepsilon > 0$. Consider $i_{v,w} > 0$. As rounding up edge weights increases the weight of an $h$-hop path additively by less than $b(i)h$, the choice of $i_{v,w}$ yields that

$$\mathrm{wd}_{i_{v,w}}(v,w) < \mathrm{wd}_h(v,w) + b(i_{v,w})h \leq (1+\varepsilon)\mathrm{wd}_h(v,w) .$$

To see the second bound, note that, by definition,

$$b(i_{v,w}) > \frac{1}{1+\varepsilon} \cdot \frac{\varepsilon\, \mathrm{wd}_h(v,w)}{h}.$$

Therefore,

$$(1+\varepsilon)\mathrm{wd}_h(v,w) < \frac{(1+\varepsilon)^2 b(i_{v,w})h}{\varepsilon},$$

and the result follows since $\varepsilon \leq 1$. □

Next, let $G_i$ be the *unweighted* graph obtained by replacing each edge $e$ in $(V, E, W_i)$ by a path of length $W_i(e)/b(i)$ (recall that $W_i(e)$ is always divisible by $b(i)$). Let $\mathrm{hd}_i(v,w)$ denote the distance between $v$ and $w$ in $G_i$. Lemma 5.2 implies that in $G_{i_{v,w}}$, the resulting hop distance between $v$ and $w$ is not too large.

**Corollary 5.3** *For all $v, w \in V$: if $\mathrm{hd}(v,w) \leq h$, then $\mathrm{hd}_{i_{v,w}}(v,w) < 4h/\varepsilon$.*

***Proof*** By Lemma 5.2, $\mathrm{wd}_{i_{v,w}}(v,w) < 4b(i_{v,w})h/\varepsilon$. As edge weights are scaled down by factor $b(i_{v,w})$ in $G_{i_{v,w}}$, we conclude that $\mathrm{hd}_{i_{v,w}}(v,w) < 4h/\varepsilon$. □

These simple observations give rise to an efficient algorithm for approximate source detection by reduction to the unweighted case.

**Theorem 5.4** *Given $0 < \varepsilon \leq 1$, any deterministic algorithm for unweighted $(S,h,\sigma)$-detection with running time $R(h,\sigma)$ can be employed to solve $(1+\varepsilon)$-approximate $(S,h,\sigma)$-estimation in $\mathcal{O}(\log_{1+\varepsilon} n \cdot R(h',\sigma) + D)$ rounds, where $h' = \lceil 4h/\varepsilon \rceil$.*

***Proof*** Let $\mathcal{A}$ be any deterministic algorithm for unweighted $(S,h,\sigma)$-detection with running time $R(h,\sigma)$. We use the following algorithm for approximate source detection.

1. For all $i \in \{0, \ldots, i_{\max}\}$, solve unweighted $(S,h',\sigma)$-detection on $G_i$ by $\mathcal{A}$. Let $L_{v,i}$ denote the output for $G_i$ at node $v$.
2. For each source $s \in S$, each node $v$ computes
$$\tilde{\mathrm{wd}}(v,s) = \inf\{\mathrm{hd}_i(v,s)b(i) \mid (\mathrm{hd}_i(v,s), s) \in L_{v,i} \text{ for some } 0 \leq i \leq i_{\max}\}.$$
3. Let $L'_v$ be the list $\{(\tilde{\mathrm{wd}}(v,s), s) \mid s \in S \text{ and } \tilde{\mathrm{wd}}(v,s) < \infty\}$, ordered in increasing lexicographical order. Node $v$ outputs $L_v = \mathrm{top}_\sigma(L'_v)$.

Clearly, the resulting running time is the one stated in the claim of the theorem.[6] In the remainder of the proof, we show correctness. First, we define

$$\mathrm{wd}'(v,s) = \inf\{\mathrm{hd}_i(v,s)b(i) \mid 0 \leq i \leq i_{\max} \text{ and } \mathrm{hd}_i(v,s) \leq h'\}.$$

We claim that $\mathrm{wd}'$ satisfies the problem specification and that the list returned by $v$ is the one induced by $\mathrm{wd}'$, which will complete the proof. The claim is established using the following properties.

(i) $\forall v \in V, s \in S : \; \mathrm{wd}'(v,s) \geq \mathrm{wd}(v,s)$,
(ii) $\forall v \in V, s \in S : \; \mathrm{wd}'(v,s) \leq (1+\varepsilon)\mathrm{wd}_h(v,s)$,
(iii) $\forall v \in V, s \in S : \; (\tilde{\mathrm{wd}}(v,s), s) \geq (\mathrm{wd}'(v,s), s)$, and
(iv) $\forall v \in V, (\tilde{\mathrm{wd}}(v,s), s) \in L_v : \; \tilde{\mathrm{wd}}(v,s) = \mathrm{wd}'(v,s)$.

We now prove these four properties.

(i) By definition,

$$b(i)\mathrm{hd}_i(v,s) = \mathrm{wd}_i(v,s) \geq \mathrm{wd}(v,s)$$

for all $v \in V$ and $s \in S$.

(ii) If $\mathrm{hd}(v,s) > h$, then $\mathrm{wd}_h(v,s) = \infty$ and the statement is trivial. Otherwise, $\mathrm{hd}(v,s) \leq h$, implying $\mathrm{hd}_{i_{v,s}}(v,s) \leq h'$ by Corollary 5.3. Hence,

$$\begin{aligned}\mathrm{wd}'(v,s) &\leq b(i_{v,s})\mathrm{hd}_{i_{v,s}}(v,s) \\ &= \mathrm{wd}_{i_{v,s}}(v,s) \\ &< (1+\varepsilon)\mathrm{wd}_h(v,s)\end{aligned}$$

[6] The additive $D$ in the running time originates in the need for nodes to determine $i_{\max}$ by learning $\max_{e \in E}\{W(e)\}$.

by Lemma 5.2.

(iii) This trivially holds, because $(\mathrm{hd}_i(v,s),s) \in L_{v,i}$ implies that $\mathrm{hd}_i(v,s) \leq h'$ (we executed $(S,h',\sigma)$-detection on each $G_i$), i.e., $\tilde{\mathrm{wd}}(v,s)$ is an infimum taken over a subset of the set used for $\mathrm{wd}'(v,s)$.

(iv) Assume for contradiction that $(\tilde{\mathrm{wd}}(v,s),s) \in L_v$, yet $\tilde{\mathrm{wd}}(v,s) > \mathrm{wd}'(v,s)$ (by the previous property $\tilde{\mathrm{wd}}(v,s) < \mathrm{wd}'(v,s)$ is not possible). Choose $i$ such that $b(i)\mathrm{hd}_i(v,s) = \mathrm{wd}'(v,s)$ and $\mathrm{hd}_i(v,s) \leq h'$. We have that $(\mathrm{hd}_i(v,s),s) \notin L_{v,i}$, as otherwise we had $\tilde{\mathrm{wd}}(v,s) \leq b(i)\mathrm{hd}_i(v,s) = \mathrm{wd}'(v,s)$. It follows that $|L_{v,i}| = \sigma$ and, for each $(\mathrm{hd}_i(v,t),t) \in L_{v,i}$, we have that

$$\begin{aligned}(\tilde{\mathrm{wd}}(v,t),t) &\leq (b(i)\mathrm{hd}_i(v,t),t)\\ &< (b(i)\mathrm{hd}_i(v,s),s)\\ &= (\mathrm{wd}'(v,s),s)\\ &\leq (\tilde{\mathrm{wd}}(v,s),s)\,,\end{aligned}$$

where in the final step we exploit the third property. As there are $\sigma$ distinct such sources $t$, we arrive at the contradiction that $(\tilde{\mathrm{wd}}(v,s),s) \notin L_v$. □

Applying Theorem 5.4 to the source detection algorithm from Sect. 4 and noting that $\log_{1+\varepsilon} n \in \Theta(\varepsilon^{-1}\log n)$ for $0 < \varepsilon \in \mathcal{O}(1)$, we obtain a variant of our second main result, Theorem 1.5, that does not rely on an a priori bound on the maximum edge weight.

**Theorem 5.5** *For $0 < \varepsilon \in \mathcal{O}(1)$, $(1+\varepsilon)$-approximate $(S,h,\sigma)$-detection can be solved in $\mathcal{O}((\varepsilon^{-1}\sigma+\varepsilon^{-2}h)\log n+D)$ rounds.*

Theorem 1.5 follows by the same arguments, if we rely on an a priori bound on $i_{\max}$ derived from a known polynomial upper bound on the maximum edge weight; then the algorithm given in the proof of Theorem 5.4 can be executed without determinining the maximum edge weight, avoiding the additive cost of $\mathcal{O}(D)$ in terms of round complexity.

## 5.2 Additional properties

As our approach is based on reduction to the unweighted case and applying Algorithm 1, the additional useful properties of the algorithm carry over.

**Corollary 5.6** *Consider augmenting the algorithm from Theorem?? so that each node $v \in V$ records the following information for each instance $i$ of the unweighted source detection:*

- *The parent of $v$ in each of the induced trees (rooted at sources).*
- *The round in which the message establishing the parent-child relation was received.*
- *The (weighted) distance in the tree to the source at which the tree is rooted.*

*Then $\tilde{\mathcal{O}}(\sigma/\varepsilon)$ bits suffice to store this extra information, and it can be used to do*

(i) *Concurrent flooding on all these trees in $\tilde{\mathcal{O}}(\varepsilon^{-1}\sigma+\varepsilon^{-2}h)$ rounds.*
(ii) *Concurrent echo on all these trees in $\tilde{\mathcal{O}}(\varepsilon^{-1}\sigma+\varepsilon^{-2}h)$ rounds.*
(iii) *Routing and distance approximation to the nodes in $L_v$ with stretch $1+\varepsilon$. The induced routing paths have $\tilde{\mathcal{O}}(\varepsilon^{-2}h)$ hops.*
(iv) *Concurrent flooding on the induced routing trees in $\tilde{\mathcal{O}}(\varepsilon^{-1}\sigma+\varepsilon^{-2}h)$ rounds.*
(v) *Concurrent echo on the induced routing trees in $\tilde{\mathcal{O}}(\varepsilon^{-1}\sigma+\varepsilon^{-2}h)$ rounds.*

***Remark*** One subtlety to be aware of is that, due to the multiple weight classes, node $v$ may have several options for the next routing hop to a given destination $w$ with an entry $(\mathrm{hd}_i(v,w),w) \in L_{v,i}$ for some $i$. In order to ensure that routing is stateless (i.e., the suffix of a routing path is independent of its prefix), nodes will always pick the next routing hop by using the entry minimizing $\mathrm{wd}_i(v,w) = \mathrm{hd}_i(v,w)b(i)$ (ties broken by choosing the smallest suitable value of $i$). This is necessary to ensure that the weight of a routing path never exceeds the distance estimate $\min_i\{\mathrm{wd}_i(v,w) \mid (\mathrm{hd}_i(v,w),w) \in L_{v,i}\}$, but implies that routing paths may have more than $h' \in \mathcal{O}(h/\varepsilon)$ hops. The bound of $\tilde{\mathcal{O}}(\varepsilon^{-2}h)$ follows from observing that for all $v,w \in V$ and $i < j$, it holds that $\mathrm{wd}_i(v,w) \leq \mathrm{wd}_j(v,w)$, and thus the $i$-value minimizing $\mathrm{wd}_i(v,w)$ is decreasing along the routing path; for each of the $\mathcal{O}(\log_{1+\varepsilon} n) \subset \tilde{\mathcal{O}}(\varepsilon^{-1})$ weight classes, the subpath for that class has $h' \in \mathcal{O}(h/\varepsilon)$ hops.

This point is also reflected in parts (iv) and (v) of Corollary 5.6: exploiting the monotonicity of routing paths with respect to $i$, the operations can be broken down into sequential (partial) flooding or echo operations for each of the $\tilde{\mathcal{O}}(\varepsilon^{-1})$ weight classes, which then each can be handled in $\mathcal{O}(\sigma+h') \subseteq \mathcal{O}(\sigma+\varepsilon^{-1}h)$ rounds.

## 5.3 Approximate source detection in directed graphs

As the reduction to the unweighted case is oblivious to whether the graph is directed or not, also our approximate source detection algorithm can be used in directed graphs. Here, we again need to consider the distance measures induced by directed paths.

**Definition 5.7** (*Directed Approximate Source Detection*) Given a weighted directed graph $G = (V, E)$, define for $v, w \in V$ that $\mathrm{hd}(v, w)$ is the minimum number of hops on a directed path from $v$ to $w$. Moreover, denote for $h \in \mathbb{N}$ by $\mathrm{wd}_h(v, w)$ the minimum weight of paths from $v$ to $w$ of at most $h$ hops (or $\infty$ if no such path exists).

Given $S \subseteq V$, $h, \sigma \in \mathbb{N}$, and $\varepsilon > 0$, let $\mathcal{L}_v^{(h,\varepsilon)}$ be a list of $\{(\mathrm{wd}'(s, v), s) \mid s \in S,\ \mathrm{wd}'(s, v) < \infty\}$, ordered in increasing lexicographical order, for some $\mathrm{wd}' : S \times V \to \mathbb{N}_\infty$ that satisfies $\mathrm{wd}'(s, v) \in \left[\mathrm{wd}(s, v),\ (1+\varepsilon)\mathrm{wd}_h(s, v)\right]$ for all $s \in S$ and $v \in V$. The directed $(1+\varepsilon)$-*approximate* $(S, h, \sigma)$-*detection* problem is to output $\mathrm{top}_\sigma(\mathcal{L}_v^{(h,\varepsilon)})$ at each node $v$ for some such $\mathrm{wd}'$.

In the simulation argument, one simply replaces undirected edges by directed edges, which does not affect the running time. If communication is still possible in both directions of each edge, this yields the following corollary.

**Corollary 5.8** *For $0 < \varepsilon \in \mathcal{O}(1)$, directed $(1+\varepsilon)$-approximate $(S, h, \sigma)$-detection can be solved in $\mathcal{O}((\varepsilon^{-1}\sigma + \varepsilon^{-2}h)\log n + D)$ rounds.*

We stress that also here, determining parameters globally, detecting termination, or other more advanced operations require bidirectional communication. If this is not possible, the above corollary does not apply, and only Theorem 1.5 applies (which assumes a known a-priori bound on the maximum edge weight).

## 6 Skeletons and skeleton spanners

In this section we define a skeleton graph $G_{\mathcal{S},h}$ of $G$, where $|\mathcal{S}|, h \in \tilde{\mathcal{O}}(\sqrt{n})$, and construct a sparse spanner of this graph. Later, we discuss approximate versions based on approximate source detection.

**Definition 6.1** (*Skeleton Graph, restated*) Let $G = (V, E, W)$ be a weighted graph. Given $\mathcal{S} \subseteq V$ and $h \in \mathbb{N}$, the *$h$-hop $\mathcal{S}$-skeleton graph* is the weighted graph $G_{\mathcal{S},h} = (\mathcal{S}, E_{\mathcal{S},h}, W_{\mathcal{S},h})$ defined by

- $E_{\mathcal{S},h} = \{\{v, w\} \mid v, w \in \mathcal{S} \wedge v \neq w \wedge \mathrm{hd}(v, w) \leq h\}$;
- For $\{v, w\} \in E_{\mathcal{S},h}$, $W_{\mathcal{S},h}(v, w) = \mathrm{wd}_h(v, w)$.

We denote the distance function in $G_{\mathcal{S},h}$ by $\mathrm{wd}_{\mathcal{S},h}$.

A simple but crucial observation on distances in skeleton graphs (which, in this context, are meant to be weighted distances) is that if the skeleton $\mathcal{S}$ consists of nodes chosen independently at random, and if $h \in \Omega(n \log n/|\mathcal{S}|)$, then w.h.p., the distances in $G_{\mathcal{S},h}$ are equal to the corresponding distances in $G$. The following lemma formalizes this idea.

**Lemma 6.2** *Let $1 \geq \pi \geq c \log n/h$ for a sufficiently large constant $0 < c \leq h/\log n$, and let $\mathcal{S}$ be a set of random nodes defined by $\Pr[v \in \mathcal{S}] = \pi$ independently for all nodes. Then w.h.p. $\mathrm{wd}_{\mathcal{S},h}(v, w) = \mathrm{wd}(v, w)$ for all $v, w \in \mathcal{S}$.*

***Proof*** Fix $v, w \in \mathcal{S}$. Clearly, $\mathrm{wd}_{\mathcal{S},h}(v, w) \geq \mathrm{wd}(v, w)$ because each path in $G_{\mathcal{S},h}$ corresponds to a path of the same weight in $G$. We show that $\mathrm{wd}_{\mathcal{S},h}(v, w) \leq \mathrm{wd}(v, w)$ as well. Let $p = \langle u_0 = v, u_1, \ldots, u_{\ell(p)} = w\rangle$ be a shortest path connecting $v$ and $w$ in $G$, i.e., $W(p) = \mathrm{wd}(v, w)$. We prove, by induction on $\ell(p)$, that $\mathrm{wd}(p) \geq \mathrm{wd}_{\mathcal{S},h}(v, w)$ w.h.p.

For the base case note that if $\ell(p) \leq h$, then by definition $\mathrm{wd}_{\mathcal{S},h}(v, w) \leq W(p) = \mathrm{wd}(v, w)$ and we are done. For the inductive step, assume that the claim holds for all values of $\ell(p) \leq i$ for some $i \geq h$ and consider a path of length $\ell(p) = i + 1$. We have

$$P[\,|\mathcal{S} \cap \{u_1, \ldots, u_i\}| = \emptyset] \leq (1-\pi)^h \leq e^{-h\pi} = e^{-c\log n} \in n^{-\Omega(c)},$$

and thus w.h.p. the intersection is non-empty. Assume that this is the case and let $u \in \{u_1, \ldots, u_i\} \cap S$. Since $p$ is a shortest path in $G$, so are $(v, \ldots, u)$ and $(u, \ldots, w)$. Both these paths are of length at most $i$, implying by the induction hypothesis that $\mathrm{wd}_{\mathcal{S},h}(v, u) \leq \mathrm{wd}(v, u)$ and $\mathrm{wd}_{\mathcal{S},h}(u, w) \leq \mathrm{wd}(u, w)$ w.h.p., respectively. Therefore $\mathrm{wd}_{\mathcal{S},h}(v, w) \leq \mathrm{wd}_{\mathcal{S},h}(v, u) + \mathrm{wd}_{\mathcal{S},h}(u, w) \leq \mathrm{wd}(v, u) + \mathrm{wd}(u, w) = W(p) = \mathrm{wd}(v, w)$ w.h.p., completing the induction. Note that the overall number of events we consider throughout the induction is in $n^{\mathcal{O}(1)}$, and since the probability of the bad events is polynomially small, the union bound allows us to deduce that the claim holds w.h.p. □

With this in mind, we fix $h = \lceil\sqrt{n}\rceil$ and sufficiently large $\pi \in \Theta(\log n/\sqrt{n})$ for Lemma 6.2 to apply to $G_{\mathcal{S},h}$ throughout this section. (Note that both can be determined in $\mathcal{O}(D)$ time.)

### 6.1 The Baswana–Sen construction

The algorithm by Baswana and Sen [9] computes a $(2k-1)$-spanner of an $n$-node graph with $\mathcal{O}(kn^{1+1/k})$ edges in expectation, in $\mathcal{O}(k)$ rounds of the **CONGEST** model.

**Definition 6.3** (*Weighted $\alpha$-Spanners*) Let $H = (V, E, W)$ be a weighted graph and $\alpha \geq 1$. An $\alpha$-spanner of $H$ is a subgraph $H' = (V, E', W')$ of $G$ where $E' \subseteq E$ and $W'$ is a restriction of $W$ to $E'$, such that $\mathrm{wd}_{H'}(u, v) \leq \alpha \cdot \mathrm{wd}_H(u, v)$ for all $u, v \in V$, where $\mathrm{wd}_H$ and $\mathrm{wd}_{H'}$ denote weighted distances in $H$ and $H'$, respectively.

We will simulate the Baswana-Sen algorithm on $G_{\mathcal{S},h}$, while running on the underlying physical graph $G$, without ever constructing the skeleton graph explicitly. Before discussing the simulation, let us recall the algorithm; we use

a slightly simpler variant that may select some additional edges, albeit without affecting the probabilistic upper bound on the number of spanner edges (cf. Lemma 6.5). The input is a graph $H = (V_H, E_H, W_H)$ and a parameter $k \in \mathbb{N}$.

1. Initially, each node is a singleton *cluster*: $R_1 := \{\{v\} \mid v \in V_H\}$.
2. For $i = 1, \ldots, k-1$ do (the $i^{th}$ iteration is called "phase $i$"):
   (a) Each cluster from $R_i$ is *marked* independently with probability $|V_H|^{-1/k}$. $R_{i+1}$ is defined to be the set of clusters marked in phase $i$.
   (b) If $v$ is a node in an unmarked cluster:
      (i) Define $Q_v$ to be the set of edges that consists of the lightest edge from $v$ to each cluster in $R_i$ it is adjacent to.
      (ii) If $v$ is not adjacent to any marked cluster, all edges in $Q_v$ are added to the spanner.
      (iii) Otherwise, let $u$ be the closest neighbor of $v$ in a marked cluster. In this case $v$ adds to the spanner the edge $\{v, u\}$, and also all edges $\{v, w\} \in Q_v$ with $(W_H(v, w), w) < (W_H(v, u), u)$ (i.e., ordered by weight, breaking ties by identifiers). Also, let $X$ be the cluster of $u$. Then $X := X \cup \{v\}$. (I.e., $v$ *joins* the cluster of $u$.)
3. Each node $v$ adds, for each cluster $X \in R_k$ it is adjacent to, the lightest edge connecting it to $X$.

For this algorithm, Baswana and Sen prove the following result.

**Theorem 6.4** [9] *Given $H = (V_H, E_H, W_H)$ and and $k \in \mathbb{N}$, the algorithm above computes a $(2k-1)$-spanner of $H$. It has $\mathcal{O}(k|V_H|^{1+1/k} \log n)$ edges w.h.p.*[7]

## 6.2 Constructing the skeleton spanner

In our case, each edge considered in Steps (2b) and (3) of the spanner algorithm on $G_{\mathcal{S},h}$ corresponds to a shortest path in $G$. Essentially, we implement these steps by letting each skeleton node find its closest $\mathcal{O}(|\mathcal{S}|^{1/k} \log n)$ clusters (w.h.p.), by running $(\mathcal{S}, h, \sigma)$-detection with $\sigma = \mathcal{O}(|\mathcal{S}|^{1/k} \log n)$. This requires a tweak: all nodes $v$ in a cluster $X$ use the same source identifier $\text{source}(v) = X$; logically, this can be interpreted as connecting them to a virtual source $X$ by edges of weight 0. Consequently, $\sigma$ needs to account for the number of detected *clusters* only, i.e., the number of nodes per cluster is immaterial. The following lemma shows that this strategy is sound.

**Lemma 6.5** *W.h.p., for a sufficiently large constant $c > 0$, execution of the centralized spanner construction algorithm yields identical results if in Steps (2b) and (3), each node considers the lightest edges to the $c \cdot |V_H|^{1/k} \log n$ closest clusters only.*

***Proof*** Fix a node $v$ and a phase $1 \leq i < k$. If $v$ has at most $c|V_H|^{1/k} \log n$ adjacent clusters, the lemma is trivially true. So suppose that $v$ has more than $c|V_H|^{1/k} \log n$ adjacent clusters. By the specification of Step (2b), we are interested only in the clusters closer than the closest marked cluster. Now, the probability that none of the closest $c|V_H|^{1/k} \log n$ clusters is marked is $(1 - |V_H|^{-1/k})^{c|V_H|^{1/k} \log n} \in n^{-\Omega(c)}$. In other words, choosing a sufficiently large constant $c$, we are guaranteed that w.h.p., at least one of the closest $c|V_H|^{1/k} \log n$ clusters is marked.

Regarding Step (3), observe that a cluster gets marked in all of the first $k-1$ iterations with independent probability $|V_H|^{-(k-1)/k}$. By Chernoff's bound, the probability that more than $c|V_H|^{1/k} \log n$ clusters remain in the last iteration is thus bounded by $2^{-\Omega(c \log n)} = n^{-\Omega(c)}$. Therefore, w.h.p. no node is adjacent to more than $c|V_H|^{1/k} \log n$ clusters in Step (3), and we are done. □

We remark that while nodes $v$ in the same cluster $X$ act as a single source, we need to keep account of the actual node $v \in X$ to which an edge in $G_{\mathcal{S},h}$ (i.e., the corresponding path in $G$) leads. This is achieved by simply adding the identifier $v$ to the messages $(d_v, X)$ of the source detection algorithm that indicate a path to $v$ and storing it alongside the respective entry of $L_v$; this does not affect the execution of the algorithm in any other way. Detailed pseudo-code of our implementation is given in Algorithm 2. Each skeleton node $s \in \mathcal{S}$ records the ID of its cluster in phase $i$ as $F_i(s)$; nodes in $V \setminus \mathcal{S}$ or those which do not join a cluster in some phase $i$ have $F_i(s) = \perp$.

To prove the algorithm correct, we argue that its executions can be mapped to executions of the centralized algorithm on the skeleton graph and then apply Theorem 6.4. This mapping is straightforward. Clusters are referred to by the identifiers of their leaders. Initially, these are the nodes sampled into $\mathcal{S}$, each of which forms a singleton cluster. The leader of a cluster in phase $i+1$ is the leader of the corresponding cluster from phase $i$ that was marked in Line 9 of iteration $i$ of the main loop of the algorithm. The broadcast in Line 5 ensures that all nodes know the cluster leaders and can decide whether $F_i(t) \in R_{i+1}$ in Line 20 locally. A call to source detection then serves to discover the skeleton edges that are added to the spanner in iteration $i$. The call uses $h = \lceil \sqrt{n} \rceil$, as we consider the $\lceil \sqrt{n} \rceil$-hop skeleton, and $\sigma \in \mathcal{O}(\mathcal{S}|^{1/k} \log n)$ suffices according to Lemma 6.5. Nodes evaluate which skeleton edges to add to the spanner locally, and update their cluster leader to the one of the closest marked cluster of this iteration. Checking for $s \neq t$ when adding spanner edges avoids adding 0-weight loops, as of course each node will determine that its own cluster is the closest source. Finally, the spanner

[7] Baswana and Sen prove that the expected number of edges is $\mathcal{O}(k|V_H|^{1+1/k})$. The modified bound directly follows from Lemma 6.5.

[8] I.e., at initialization of Algorithm 1 set $L_v := \{(0, F_i(v))\}$ if $F_i(v) \neq \perp$ and $L_v := \emptyset$ otherwise.

**Algorithm 2:** Construction of skeleton spanner.

**input** : `// trades approximation for sparsity` $k$: integer in $[1, \log n]$
**output**: $\mathcal{S} \subseteq \mathcal{V}$ `// skeleton nodes` $E_{\mathcal{S},h,k} \subseteq V$ `// skeleton spanner edges` $W_{\mathcal{S},h,k} : E_k \to \mathbb{N}$ `// edge weights`

1 $\mathcal{S} := \emptyset$
2 $E_{\mathcal{S},h,k} := \emptyset$
3 **foreach** $v \in V$ **do**
  `// c is a sufficiently large constant`
4  add $v$ to $\mathcal{S}$ with probability $c \log n / \sqrt{n}$
  $F_1(v) := \begin{cases} v & \text{if } v \in \mathcal{S} \\ \bot & \text{otherwise} \end{cases}$
5 broadcast $\mathcal{S}$ to all nodes
`// cluster leaders; initial clusters are singletons of` $\mathcal{S}$
6 $R_1 := \mathcal{S}$
`// c is a sufficiently large constant`
7 $\sigma := c \cdot |\mathcal{S}|^{1/k} \log n$ **for** $i := 1$ **to** $k$ **do**
8  **if** $i < k$ **then**
9  $R_{i+1} :=$ random subset of $R_i$ of expected size $|\mathcal{S}|^{1-i/k} = |R_i|/|\mathcal{S}|^{1/k}$
  `// make leaders of marked clusters known`
10  broadcast $R_{i+1}$ to all nodes
11  **else**
  `// no clusters marked in final iteration`
12  $R_{i+1} := \emptyset$
13  solve $(\mathcal{S}, \lceil\sqrt{n}\rceil, \sigma)$-detection on $G$, using source identifier $F_i(v)$ at $v$;[8] record the node $w$ for each entry $(d, F_i(w)) \in L_v$
14  **foreach** $s \in \mathcal{S}$ **do**
15  Let $L_s$ denote the list returned by the call to $(\mathcal{S}, \lceil\sqrt{n}\rceil, \sigma)$-detection
16  $F_{i+1}(s) := \bot$
17  **foreach** $(\mathrm{wd}(s,t), F_i(t)) \in L_s$ *in increasing lexicographical order* **do**
18  **if** $s \neq t$ **then**
  `// add edge to spanner`
19  $E_{\mathcal{S},h,k} := E_{\mathcal{S},h,k} \cup \{s,t\}$ $W_{\mathcal{S},h,k} := \mathrm{wd}(s,t)$
20  **if** $F_i(t) \in R_{i+1}$ **then**
  `// leader of closest marked cluster`
21  $F_{i+1}(s) := F_i(t)$
22  **break**
23 broadcast $E_{\mathcal{S},h,k}$, and $W_{\mathcal{S},h,k}$ to all nodes
24 **return** $(\mathcal{S}, E_{\mathcal{S},h,k}, W_{\mathcal{S},h,k})$

is made known to all nodes by broadcasting it over a BFS tree.

**Lemma 6.6** *W.h.p., Algorithm 2 can be implemented with the following guarantees.*

(i) $|\mathcal{S}| \in \Theta(n^{1/2} \log n)$.

(ii) *It computes a weighted* $(2k-1)$*-spanner of the skeleton graph* $G_{\mathcal{S},\lceil\sqrt{n}\rceil}$ *that is known at all nodes and has* $\tilde{\mathcal{O}}(n^{1/2+1/(2k)})$ *edges.*

(iii) *The weighted distances between nodes in* $\mathcal{S}$ *are identical in* $G_{\mathcal{S},\lceil\sqrt{n}\rceil}$ *and* $G$.

(iv) *The algorithm terminates in* $\tilde{\mathcal{O}}(n^{\frac{k+1}{2k}} + D)$ *rounds.*

***Proof*** Statement (i) is immediate from an application of Chernoff's bound, as each node joins $\mathcal{S}$ independently with probability $\Theta(\log n/\sqrt{n})$. To prove Statement (ii), we note that Algorithm 2 simulates the centralized algorithm, except for considering only the closest $\mathcal{O}(|\mathcal{S}|^{1/k} \log n)$ clusters when adding edges to the spanner. By Lemma 6.5, this results in a (simulated) correct execution of the centralized algorithm w.h.p. Hence, Statement (ii) follows from Theorem 6.4 and Statement (i). Statement (iii) follows from Lemma 6.2.

It remains to analyze the running time of the algorithm. All steps but the broadcast operations (Lines 5, 10, and 23) and the call to source detection (Line 13) are local computations. Lemma 3.3 together with Statements (i) and (ii) implies that the broadcast operations can be completed within $\tilde{\mathcal{O}}(n^{1/2+1/(2k)} + D)$ rounds in total. (Note that $k$ factors are absorbed in the weak $\tilde{\mathcal{O}}$ notation because $k \leq \log n$.) Source detection can be solved in $\mathcal{O}(\sigma h)$ rounds [32]. As $h = \lceil\sqrt{n}\rceil$ and, by Statement (i), $\sigma \in \tilde{\mathcal{O}}(n^{1/(2k)})$, the time complexity bound follows. □

We remark that it is not difficult to derandomize the algorithm at the cost of a multiplicative increase of $\mathcal{O}(\log n)$ in the running time, see [10].

### 6.3 Routing on the skeleton spanner

Algorithm 2 constructs a $(2k-1)$-spanner of the skeleton graph and makes it known to all nodes. This enables each skeleton node to determine low-stretch routing paths in $G_{S,h}$ by local computation. To use this information, we must map each spanner edge $e = \{s,t\} \in E_{S,h}$ to a path in $G$ of weight $W_{S,h}(s,t)$. Since the construction of the spanner was carried out by source detection, we can readily map a spanner edge to a route in $G$ in one direction: if, say, $s$ added the edge $\{s,t\}$ to the spanner, then that edge corresponds to a path in the induced tree (of depth at most $h$) rooted at $t$, which can be easily reconstructed using the weight information, thus facilitating routing from $s$ to $t$. However, to route in the opposite direction we need to do a little more.[9] Specifically, we add a post-processing step where we "reverse" the unidirectional routing paths, i.e., inform the nodes on the paths about their predecessors (if we have paths both from $s$ to $t$ and vice versa,

[9] This asymmetry is not due to our implementation: consider an $n$-node star graph. Its $k$-spanner is the whole star (for any $k \geq 1$). However, the center adds only $\tilde{\mathcal{O}}(n^{1/k})$ edges to the spanner.

we select one to reverse and drop the other). This can be done in $\mathcal{O}(\sigma h)$ rounds by using the idea in Corollary 4.7, part (iv).

**Corollary 6.7** *Let $e = \{s,t\}$ be a skeleton spanner edge selected by Algorithm 2. Denote by $p_e \in \text{paths}(s,t)$ the corresponding path in $G$ of $\ell(p_e) \leq h$ hops and weight $W(p_e) = \text{wd}_h(s,t) = W_{\mathcal{S},h}(s,t)$ that was (implicitly) found by the call to source detection when the edge was added. Then, concurrently for all $e \in E_{\mathcal{S},h}$, each node $v$ on $p_e$ can learn the next nodes on this path in both directions within $\tilde{\mathcal{O}}(n^{\frac{k+1}{2k}})$ rounds w.h.p.*

Our third main result, Theorem 1.7, now follows from Lemma 6.6 and Corollary 6.7.

### 6.4 Approximate skeleton and skeleton spanner

The reduction of the single-source shortest path problem to an overlay network on $\tilde{\mathcal{O}}(\sqrt{n})$ nodes given in [24] is based on computing approximate distances to the source on a skeleton. However, this requires the skeleton to be known as an overlay network, which means that its nodes have knowledge of their incident edges. We illustrated in Fig. 3 why an algorithm obtaining this information cannot be fast. However, using approximate source detection, we can compute an "approximate" skeleton graph.

**Definition 6.8** (*Approximate Skeleton Graph*) Let $G = (V, E, W)$ be a weighted graph. Given $\mathcal{S} \subseteq V$ and $h \in \mathbb{N}$, a $(1+\varepsilon)$-*approximate* $h$-hop $\mathcal{S}$-*skeleton graph* is a weighted graph $\tilde{G}_{\mathcal{S},h} = (\mathcal{S}, E_{\mathcal{S},h}, \tilde{W}_{\mathcal{S},h})$ satisfying

- $E_{\mathcal{S},h} = \{\{v,w\} \mid v,w \in \mathcal{S} \wedge v \neq w \wedge \text{hd}(v,w) \leq h\}$;
- For $\{v,w\} \in E_{\mathcal{S},h}$, $\text{wd}_h(v,w) \leq \tilde{W}_{\mathcal{S},h}(v,w) \leq (1+\varepsilon)\text{wd}_h(v,w)$.

We denote the distance function in $\tilde{G}_{\mathcal{S},h}$ by $\tilde{\text{wd}}_{\mathcal{S},h}$.

Recall that, for sufficiently large $h$, an (exact) skeleton on independently sampled nodes preserves distances w.h.p. Analogously, a $(1+\varepsilon)$-approximate skeleton preserves distances up to factor $1+\varepsilon$.

**Corollary 6.9** *For a given parameter $h \in \mathbb{N}$, let $\mathcal{S}$ be a set of nodes obtained by adding each node from $V$ independently with probability $\pi \geq c \log n/h$, where $0 < c \leq h/\log n$ is a sufficiently large constant. Let $\tilde{G}$ be any $(1+\varepsilon)$-approximate $h$-hop $\mathcal{S}$-skeleton of $G$ for a given parameter $\varepsilon > 0$. Then w.h.p. (over the choice of $\mathcal{S}$), for all $v,w \in \mathcal{S}$ we have $\text{wd}(v,w) \leq \tilde{\text{wd}}_{\mathcal{S},h}(v,w) \leq (1+\varepsilon)\text{wd}(v,w)$.*

***Proof*** As for Lemma 6.2, taking into account that $h$-hop distances are only approximated up to factor $1+\varepsilon$. □

Using approximate source detection, we can compute an approximate skeleton, in the sense that each skeleton node learns its incident edges and their weights.

**Corollary 6.10** *Let $\mathcal{S}$ and $h$ be as in Corollary 6.9 and $0 < \varepsilon \in \mathcal{O}(1)$. We can compute a $(1+\varepsilon)$-approximate $h$-hop $\mathcal{S}$-skeleton of $G$ in $\tilde{\mathcal{O}}(\varepsilon^{-1}|\mathcal{S}| + \varepsilon^{-2}h + D)$ rounds.*

***Proof*** After determining $|\mathcal{S}|$ in $\mathcal{O}(D)$ rounds, we run $(1+\varepsilon)$-approximate $(\mathcal{S}, h, |\mathcal{S}|)$-detection, which by Theorem 5.4 completes within the stated time bounds. Note, however, that the distance estimates nodes $s,t \in \mathcal{S}$ have obtained from each other may differ. To fix this, we leverage Statement (i) of Corollary 5.6, "reversing" the flow of distance information as compared to the algorithm, again taking $\tilde{\mathcal{O}}(\varepsilon^{-1}|\mathcal{S}|+\varepsilon^{-2}h)$ rounds. As a result, $s$ will obtain the estimate $t$ has of its distance to $s$ and vice versa. Now each skeleton edge is assigned the minimum of the two values as weight. □

Given the information obtained in the construction of the overlay, one can readily run the Baswana-Sen algorithm on the overlay to obtain a spanner of the approximate skeleton.

**Corollary 6.11** *For any integer $k \in [1, \log n]$, w.h.p. we can compute and make known to all nodes a $(2k-1)$-spanner of the approximate skeleton determined in Corollary 6.10 of $\tilde{\mathcal{O}}(|\mathcal{S}|^{1+1/k})$ edges within $\tilde{\mathcal{O}}(|\mathcal{S}|^{1+1/k} + D)$ additional rounds.*

We remark that [10,24] provide derandomizations, resulting in a deterministic $(1+o(1))$-approximation to SSSP distances within $\tilde{\mathcal{O}}(\sqrt{n}+D)$ rounds.

For later use in our routing schemes we specialize the result as follows.

**Corollary 6.12** *For any $0 < \varepsilon \in \mathcal{O}(1)$ and any integer $k \in [1, \log n]$, within $\tilde{\mathcal{O}}(\varepsilon^{-2}n^{\frac{2k+1}{4k}} + D)$ rounds a graph $G_{\mathcal{S}} = (\mathcal{S}, E_{\mathcal{S}}, W_{\mathcal{S}})$ with the following properties can be computed and made known to all nodes w.h.p.*

(i) *Nodes are sampled independently into $\mathcal{S}$, so that $|\mathcal{S}| \in \Theta(n^{\frac{2k-1}{4k}} \log n)$.*
(ii) *$|E_{\mathcal{S}}| \in \tilde{\mathcal{O}}(n^{\frac{2k+1}{4k}})$.*
(iii) *For all $s,t \in \mathcal{S}$, $\text{wd}(s,t) \leq \text{wd}_{\mathcal{S}}(s,t) \leq (1+\varepsilon)(2k-1)\text{wd}(s,t)$, where $\text{wd}_{\mathcal{S}}$ is the distance metric induced by $W_{\mathcal{S}}$.*

***Proof*** Choose sampling probability $\pi = n^{\frac{-2k-1}{4k}} \log n$, pick $h = c \log n/\pi \in \tilde{\mathcal{O}}(n^{\frac{2k+1}{4k}})$, and apply Corollary 6.9, Corollary 6.10, and Corollary 6.11. □

Regarding the mapping of edges in $G_{\mathcal{S}}$ to paths in $G$, we have the following.

**Corollary 6.13** *For each edge $e = \{s,t\} \in E_{\mathcal{S}}$ as in Corollary 6.12, let $p_e \in \text{paths}(s,t)$ denote its corresponding path in $G$. Then, after $\tilde{\mathcal{O}}(\varepsilon^{-2}n^{\frac{2k+1}{4k}} + D)$ additional rounds, w.h.p., every node $v$ on every path $p_e$ knows the next nodes on this path in both directions (including the weight of the respective subpaths).*

To prove this corollary, we use the powerful tool of *labeling schemes*. A *tree labeling scheme* is an assignment of labels to tree nodes such that determining the next hop from one node towards another, or the distance between two nodes, can be done based on the labels of the two nodes alone. We note that determining the next hop can be achieved with $\mathcal{O}(\log n)$-bit labels [50], while determining the distance requires $\Theta(\log^2 n)$-bit labels [21,40]. We shall use the following result, which is implicit in the work by Thorup and Zwick (see Section 2.1 and Theorem 2.6 in [52]).

**Theorem 6.14** (based on [52]) *It is possible to construct a tree labeling scheme with $\mathcal{O}(\log n)$-bit tables and $\mathcal{O}(\log^2 n)$-bit labels using $\mathcal{O}(\log n)$ flooding/echo operations in the **CONGEST** model.*

***Proof of Corollary 6.13*** In each iteration of the Baswana-Sen construction, nodes may add at most $\sigma \in \tilde{\mathcal{O}}(|\mathcal{S}|^{1/k})$ edges corresponding to their $\sigma$ closest clusters to the spanner. By Corollary 5.6 *(iv),(v)*, we can perform concurrent flooding and echo operations on the corresponding routing trees in $\tilde{\mathcal{O}}(\varepsilon^{-\frac{2k+1}{k}} n^{\frac{2k+1}{4k}})$ rounds w.h.p. Therefore, by Theorem 6.14, we can construct tree labels of $\mathcal{O}(\log^2 n)$ bits. To get rid of the labels and let each node acquire full information on the paths $p_e$ corresponding to edges $e \in E_{\mathcal{S}}$, each skeleton node $s \in \mathcal{S}$ announces the tree labels for its tree $T_s$ and for each other tree $T_t$ such that $\{s,t\} \in E_{\mathcal{S}}$. Using a BFS tree, this takes $\mathcal{O}(\sigma|\mathcal{S}|+D) \subseteq \tilde{\mathcal{O}}(\varepsilon^{-2} n^{\frac{2k+1}{4k}} + D)$ rounds w.h.p. Since for each edge $e = \{s,t\} \in E_{\mathcal{S}}$, we have that $p(e) \in T_s$ or $p(e) \in T_t$, each node can determine whether it is in $p(e)$ and, if so, its neighbors in $p(e)$ in direction of $s$ and $t$, respectively. □

# 7 Table construction in unweighted graphs

## 7.1 Exact tables

As a warm-up, let us state the following immediate result.

**Corollary 7.1** *On unweighted graphs, name-independent tables for exact (i.e., stretch-1) routing and distances can be computed in $n + \mathcal{O}(D)$ rounds.*

***Proof*** Using a BFS tree, find a bound on the diameter $d \in \mathcal{O}(D)$ and the number of nodes $n$ (cf. Sect. 3.4). Then run source detection with $S = V$, $\sigma = n$ and $h = d$. The result follows from Theorem 1.2. □

## 7.2 Tables of Size $\tilde{\mathcal{O}}(n^{1/k})$ and Stretch $4k - 3$

While Corollary 7.1 merely reproduces earlier results (albeit with improved leading constants), the fact that we solve source detection in unweighted graphs in $\sigma + h$ rounds irrespectively of $|S|$ permits efficient distributed construction of a Thorup–Zwick routing hierarchy [52].

### 7.2.1 Algorithm

Assume that $n$ and $D$ are known (cf. Sect. 3.4). Let $k \in [1, \log n]$ be an integer parameter ($k$ controls the trade-off between table size and maximum stretch). The construction algorithm is as follows.

1. Define $S_0 = V$. Given $S_{i-1}$, construct $S_i$, for $i \in \{1, \ldots, k-1\}$, by independently including each member of $S_{i-1}$ in $S_i$ with probability $n^{-1/k}$. Set $S_k = \emptyset$.
2. For $i = 0, \ldots, k-1$, run $(S_i, D, \sigma)$-detection, where $\sigma := cn^{1/k} \log n$ for a sufficiently large constant $c$. Let $T_{s,i}$ denote the induced tree for source $s \in S_i$.
3. For each tree $T_{s,i}$, construct a tree labeling scheme as in Theorem 6.14. The result, for each $v \in T_{s,i}$, is a label $\lambda_i(v)$ and a routing table $R_{s,i}(v)$ of $\mathcal{O}(\log n)$ bits, facilitating routing in $T_{s,i}$.
4. Let $s_{v,i}$ be the node in $S_i$ minimizing $\mathrm{hd}(v, s_{v,i})$. The output of a node $v \in V$ consists of (i) its label $\lambda(v)$ constructed from the ID of $v$ and the list of pairs $\{(s_{v,i}, \lambda_i(v))\}_{i=1}^{k-1}$, and (ii) a table containing the lists $L_{v,i}$ constructed by source detection[10] and the routing table $R_{s,i}(v)$ for each $s, i$ for which $(\mathrm{hd}(v,s), s) \in L_{v,i}$.

Routing proceeds as follows. Let $v$ be any node, and suppose $v$ is given the label $\lambda(w)$ of the destination $w$. Then $v$ determines the next routing hop to $w$ as follows. If $(\mathrm{hd}(v,w), w) \in L_{v,0}$, exact routing is given by $T_{w,0}$. Otherwise, $v$ finds the minimum $i \in \{0, \ldots, k-1\}$ so that $v \in T_{w_{s,i},i}$ and reports back the next routing hop from $v$ to $w$ in $T_{w_{s,i},i}$. Note that this rule does not rely on prior routing decisions, i.e., it is stateless. Distance approximation is done using the same mechanism.

### 7.2.2 Analysis

The following lemma is an immediate consequence of Chernoff's bound.

**Lemma 7.2** *In the above algorithm, we have, w.h.p., that (i) $|S_{k-1}| \leq \sigma$; and (ii) for all $v \in V$ and $i \in \{0, \ldots, k-2\}$, $\exists s \in \mathcal{S}_{i+1}$ satisfying $(\mathrm{hd}(v,s), s) \in L_{v,i}$.*

Note that part (i) of the lemma implies that for any $v, w \in V$, there is an index $i$ so that $v \in T_{w_{s,i},i}$, and hence the routing scheme is correct.

The tables and labels, by construction, are of size $\tilde{\mathcal{O}}(n^{1/k})$ and $\mathcal{O}(k \log n)$ bits, respectively. The round complexity of the construction can be readily bounded using our previous results.

[10] Including the respective parents in the induced trees; we will refrain from repeating this every time in what follows.

**Corollary 7.3** *The above algorithm runs in* $\tilde{\mathcal{O}}(n^{1/k} + D)$ *rounds.*

***Proof*** Steps 1 and 4 involve local computations only. By Theorem 1.2, Step 2 takes $\mathcal{O}(k(\sigma + D)) \subset \tilde{\mathcal{O}}(n^{1/k} + D)$ rounds. By Theorem 6.14, constructing the labels and tables in Step 3 can be performed by $\tilde{\mathcal{O}}(1)$ flooding and echo operations on each of the trees. By Corollary 4.7, these operations can be executed for each level $i$ concurrently within $\tilde{\mathcal{O}}(\sigma + D) \subset \tilde{\mathcal{O}}(n^{1/k} + D)$ rounds. □

Finally, we prove that the resulting stretch is at most $4k - 3$. We follow [52], but in our case (since the table of the destination node is not available), each step of the induction contains an additional application of the triangle inequality. Consequently the stretch is $4k - 3$ rather than $2k - 1$.

**Lemma 7.4** *Let* $v, w \in V$ *and* $1 \leq j \leq k - 1$. *If* $v \notin T_{s_{w,j}}$, *then w.h.p., (a)* $\mathrm{hd}(v, s_{v,j}) \leq (2j - 1)\mathrm{hd}(v, w)$, *and (b)* $\mathrm{hd}(w, s_{w,j}) \leq 2j \cdot \mathrm{hd}(v, w)$.

***Proof*** We show, by induction on $i \in \{1, \ldots, j\}$, that (a) $\mathrm{hd}(v, s_{v,i}) \leq (2i - 1)\mathrm{hd}(v, w)$ and (b) $\mathrm{hd}(w, s_{w,i}) \leq 2i \cdot \mathrm{hd}(v, w)$. For the basis of the induction, consider $i = 0$. In this case, since $S_0 = V$, we have that, $s_{u,0} = u$ for all nodes $u$ and the claim is trivial.

For the inductive step, assume that (b) holds for $0 \leq i < j$ and consider $i + 1$. By assumption, $v \notin T_{s_{w,i},i}$, i.e., $(\mathrm{hd}(v, s_{w,i}), s_{w,i}) \notin L_{v,i}$. However, by Statement (ii) of Lemma 7.2, $(\mathrm{hd}(v, s_{v,i+1}), s_{v,i+1}) \in L_{v,i}$ w.h.p. Hence,

$$\begin{aligned}\mathrm{hd}(v, s_{v,i+1}) &\leq \mathrm{hd}(v, s_{w,i}) \\ &\leq \mathrm{hd}(v, w) + \mathrm{hd}(w, s_{w,i}) \\ &\leq (2i + 1)\mathrm{hd}(v, w),\end{aligned}$$

where in the last step we use the induction hypothesis. This proves part (a) of the claim for index $i + 1$. As $s_{w,i+1}$ is the closest node from $S_{i+1}$ to $w$, using the above inequality we also obtain

$$\begin{aligned}\mathrm{hd}(w, s_{w,i+1}) &\leq \mathrm{hd}(w, s_{v,i+1}) \\ &\leq \mathrm{hd}(w, v) + \mathrm{hd}(v, s_{v,i+1}) \\ &\leq (2i + 2)\mathrm{hd}(v, w),\end{aligned}$$

which proves part (b) of the claim, completing the inductive step. □

Rephrasing, we obtain the following result.

**Corollary 7.5** *Let* $v, w \in V$, *and let* $0 \leq i_0 \leq k - 1$ *be minimal such that* $v \in T_{s_{w,i_0},i_0}$. *Then* $\mathrm{wd}(v, s_{w,i_0}) + \mathrm{wd}(s_{w,i_0}, w) \leq (4i_0 + 1)\mathrm{wd}(v, w) \leq (4k - 3)\mathrm{wd}(v, w)$).

***Proof*** By Lemma 7.4,

$$\begin{aligned}\mathrm{wd}(v, s_{w,i_0}) + \mathrm{wd}(s_{w,i_0}, w) &\leq \mathrm{wd}(v, w) + 2\mathrm{wd}(w, s_{w,i_0}) \\ &\leq (4i_0 + 1)\mathrm{wd}(v, w) \\ &\leq (4k - 3)\mathrm{wd}(v, w)\end{aligned}$$

and the corollary is proved. □

To summarize, we arrive at the following theorem.

**Theorem 7.6** *Given an unweighted graph and an integer* $k \in [1, \log n]$, *we can compute in* $\mathcal{O}(n^{1/k} + D)$ *rounds* $\tilde{\mathcal{O}}(n^{1/k})$*-bit tables and* $\mathcal{O}(k \log n)$*-bit labels which facilitate, w.h.p., stateless* $(4k-3)$*-stretch routing and distance approximation.*

We note that we can obtain stretch $2k - 1$ at the cost of increasing the label size to $\tilde{\mathcal{O}}(n^{1/k})$: simply append the destination's table to its label.

## 8 Table construction in weighted graphs

In this section, we use approximate source detection and skeleton spanners for constructing tables for weighted graphs. We first consider the case where the Shortest-Path Diameter (SPD, cf. Sect. 3.3) is small.

### 8.1 Small shortest-path diameter

If the SPD is small, then, intuitively, we do not need to construct a skeleton (whose role is to split shortest paths with many hops into few-hops subpaths), and we can directly apply the strategy for the unweighted case using SPD instead of $D$. However, this approach raises two issues. First, it is not known how to compute—or approximate—SPD efficiently. Second, source detection has time complexity $\tilde{\Theta}(h\sigma)$ in general, resulting in a multiplicative running time overhead of $\tilde{\Theta}(n^{1/k})$ for tables of stretch $4k - 3$.

We can solve each of these concerns, but we do not know whether one can construct tables of stretch $\Theta(k)$ in $\tilde{\mathcal{O}}(n^{1/k} + \mathrm{SPD})$ rounds. In order to obtain an algorithm that requires no initial knowledge on SPD, one can exploit the fact that for $h \geq \mathrm{SPD}$, source detection is solved if and only if each node knows the exact distance to its $\sigma$ closest sources, which holds at a round if and only if no node $v$ changes its $L_v$ list in that round. The latter property, and hence global termination, can be detected (by means similar to the ones used to prove Lemma 3.2) in $\mathcal{O}(D) \subseteq \mathcal{O}(\mathrm{SPD})$ additional rounds. We therefore have the following.

**Corollary 8.1** *For any natural* $k \in [1, \log n]$, *tables of size* $\tilde{\mathcal{O}}(n^{1/k})$ *and labels of size* $\mathcal{O}(k \log n)$ *for routing and distance approximation with stretch* $4k - 3$ *can be computed in* $\mathcal{O}(n^{1/k}\mathrm{SPD})$ *rounds w.h.p.*

***Proof*** (sketch) We use the algorithm described in Sect. 7.2, replacing the invocations of source detection in step 2 with approximate source detection using infinity as the hop bound, in conjuction with termination detection as discussed above. We observe that the stretch bound can be shown analogously to Lemma 7.4, by replacing hd with wd. □

## 8.2 The general case

If SPD is large or unknown, the algorithms outlined above may be too slow. Our approach is to use approximate source detection and a skeleton spanner.

### 8.2.1 Algorithm

We first describe a stateful routing variant (i.e., the next hop may be a function of traversed hops); we extend it to a stateless one later. The routing table computation algorithm takes $0 < \varepsilon \leq 1$ as a parameter and proceeds as follows.

1. Construct an (approximate) skeleton spanner $G_{\mathcal{S}} = (\mathcal{S}, E_{\mathcal{S}}, W_{\mathcal{S}})$ and make it known to all nodes (Corollary 6.12). Node $v \in V$ also stores the solution $L_v(\mathcal{S})$ to $(1+\varepsilon)$-approximate $(\mathcal{S}, h, |\mathcal{S}|)$-detection, where $h = n^{\frac{2k+1}{4k}}$, which is computed during the construction, as well as the routing information for $(1+\varepsilon)$-stretch routing to the detected nodes (computed using Corollary 5.6).
2. Construct a routing path $p_e$ in $G$ for each edge in $e \in E_{\mathcal{S}}$ (Corollary 6.13).
3. Run $(1+\varepsilon)$-approximate $(V, h, h)$-detection, obtaining a list $L_v(V)$ for each $v \in V$ (Theorem 1.5). Determine the necessary information to route from $v$ to $w$ with stretch $1+\varepsilon$, for each $v, w \in V$ such that $(\mathrm{wd}'(v,w), w) \in L_v(V)$ (Corollary 5.6).
4. For each $v \in V$, let $s'_v$ be the closest node of $\mathcal{S}$ w.r.t. $\mathrm{wd}'$, i.e., $(\mathrm{wd}'(v, s'_v), s'_v)$ is the first entry of $L_v(V)$ with $s'_v \in \mathcal{S}$.[11] For each $s \in \mathcal{S}$, let $T_s$ be the tree defined by the union of all routing paths from nodes $v$ with $s'_v = s$. Using Corollary 4.7 and Theorem 6.14, compute tree labels $\lambda_{v,s}$ as in [52] in each such tree $T_s$ for each $v \in T_s$. The label of node $v$ is $\lambda_v := (v, s'_v, \mathrm{wd}'(v, s'_v), \lambda_{v,s'_v})$ and its routing table contains all that was computed in the previous steps.

Routing and distance approximation is done as follows. Given the label $\lambda_w$ of $w \in V$ at node $v \in V$, $v$ checks whether there is an entry $(\mathrm{wd}'(v,w), w) \in L_v(V)$ with $\mathrm{wd}'(v,w) \leq \mathrm{wd}'(w, s'_w)$. If there is one, $v$ can estimate the distance to $w$ as $\mathrm{wd}'(v,w)$ and it knows the next hop on the corresponding route to $w$. Otherwise, $v$ estimates the distance as $\min_{s \in \mathcal{S}}\{\mathrm{wd}'(v,s) + \mathrm{wd}_{\mathcal{S}}(s, s'_w) + \mathrm{wd}'(w, s'_w)\}$, where $\mathrm{wd}_{\mathcal{S}}$ is the distance metric on $G_{\mathcal{S}}$ (using the list $L_v(\mathcal{S})$, its knowledge of $G_{\mathcal{S}}$, and the label $\lambda_w$). If a message needs to be routed, $v$ picks the next hop on the corresponding path; by adding the sequence of nodes in $\mathcal{S}$ that are still to be visited to the message, [12] each intermediate node on the path can determine its next routing hop in $G$. The weight of the routing path is bounded from above by the distance estimate computed by $v$.

[11] We slightly abuse notation here in that we do not indicate whether the distance function $\mathrm{wd}'$ corresponds to the lists $L_v(\mathcal{S})$ or $L_v(V)$ explicitly. Unless explicitly indicated otherwise, we refer to both instances.

### 8.2.2 Analysis

Due to the choice of $h = (c \log n)/\mathbb{E}(|\mathcal{S}|)$, with probability $1 - n^{-\Theta(c)}$, there is some $s \in \mathcal{S}$ such that $(\mathrm{wd}'(v,s), s) \in L_v(V)$ (cf. Corollary 6.12). Thus, w.h.p. all steps of the algorithm can be executed as described. Based on the information computed (and stored) by $v$ and the label $\lambda_w$, $v$ can always determine the above distance estimate. With the additional information included in the routing message (i.e., the subpath to take in $G_{\mathcal{S}}$), nodes can determine the next routing hop.

Concerning the round complexity, recall that tree labelings can be constructed using $\tilde{\mathcal{O}}(1)$ flooding/echo operations by Theorem 6.14. Hence, by Theorem 1.5 and Corollaries 5.6, 6.12, and 6.13, the scheme can be implemented in $\tilde{\mathcal{O}}(\varepsilon^{-2} n^{\frac{2k+1}{4k}} + D)$ rounds w.h.p.

It remains to prove that the scheme guarantees, w.h.p., stretch at most $(1+\mathcal{O}(\varepsilon))(4k-1)$. To this end, we first show that for close-by nodes, $\mathrm{wd}'$ actually approximates the real distances well. The key observation is simple: the internal nodes on any shortest path from $v$ to $w$ are closer to $v$ than $w$, and therefore if $w$ is among the closest $h+1$ nodes to $v$, then $\mathrm{wd}_h(v,w) = \mathrm{wd}(v,w)$.

**Lemma 8.2** *Fix $v$ and order $V$ in increasing lexicographical order of $(\mathrm{wd}(v,w), w)$. Let $w_1, \ldots, w_n$ be the resulting node sequence. Then $\mathrm{wd}_h(v, w_i) = \mathrm{wd}(v, w_i)$ for $i \leq h+1$.*

***Proof*** For any $i \leq h+1$, choose a shortest path $p \in \mathrm{paths}(v, w_i)$, i.e., $W(p) = \mathrm{wd}(v, w_i)$. All nodes $u \in p \backslash \{w_i\}$ satisfy that $\mathrm{wd}(v,u) < W(p) = \mathrm{wd}(v,w)$, because edge weights are positive and there is a strict subpath of $p$ connecting $v$ and $u$. We conclude that $\ell(p) \leq h$ and therefore $\mathrm{wd}_h(v, w_i) = \mathrm{wd}(v, w_i)$. □

Applying Lemma 8.2, we relate $\mathrm{wd}'$ and wd for close-by nodes.

**Corollary 8.3** *Given $v \in V$, let $s_v \in \mathcal{S}$ and $s'_v \in \mathcal{S}$ denote the skeleton nodes minimizing $(\mathrm{wd}(v,s), v)$ and $(\mathrm{wd}'(v,s), v)$, respectively. Suppose $\mathrm{wd}'$ is the distance function of an instance of $(1+\varepsilon)$-approximate $(S, h, \sigma)$-detection for any $\sigma$ and $S \supseteq \mathcal{S}$, and $\mathcal{S}$ and $h$ as in the above algorithm. Then, w.h.p.,*

[12] We ignore message size for the moment, as making the scheme stateless will remove this header.

(i) $\forall w \in V : (\mathrm{wd}(v,w),w) \leq (\mathrm{wd}(v,s_v),s_v) \Rightarrow \mathrm{wd}_h(v,w) = \mathrm{wd}(v,w)$;
(ii) $\forall w \in V : \mathrm{wd}'(v,w) \leq \mathrm{wd}'(v,s_v) \Rightarrow \mathrm{wd}'(v,w) \leq (1+\varepsilon)\mathrm{wd}(v,w)$;
(iii) $\forall w \in V : \mathrm{wd}(v,w) < \mathrm{wd}'(v,s'_v)/(1+\varepsilon) \Rightarrow \mathrm{wd}_h(v,w) = \mathrm{wd}(v,w)$.

***Proof*** Recall that nodes have been sampled into $\mathcal{S}$ with uniform and indepent probability $c \log n/h$ (for a sufficiently large constant $c$). Using the notation of Lemma 8.2, the probability that $s_v \neq w_i$ for all $i \leq h$ equals

$$\left(1 - \frac{c\log n}{h}\right)^h \in e^{-\Theta(c\log n)} = n^{-\Theta(c)}.$$

In other words, $s_v = w_i$ for some $i \leq h$ w.h.p., implying (i) by Lemma 8.2.

To show (ii), we distinguish between two cases. If $\mathrm{wd}(v,w) < \mathrm{wd}(v,s_v)$, then w.h.p., $\mathrm{wd}_h(v,w) = \mathrm{wd}(v,w)$ by (i). In this case, the claim follows directly from the properties of source detection, namely that $\mathrm{wd}'(v,w) \leq (1+\varepsilon)\mathrm{wd}_h(v,w)$. Otherwise, $\mathrm{wd}(v,w) \geq \mathrm{wd}(v,s_v)$ and we can bound

$$\begin{aligned}\mathrm{wd}'(v,w) &\leq \mathrm{wd}'(v,s_v)\\ &\leq (1+\varepsilon)\mathrm{wd}(v,s_v)\\ &\leq (1+\varepsilon)\mathrm{wd}(v,w)\,.\end{aligned}$$

For (iii), observe that

$$\frac{\mathrm{wd}'(v,s'_v)}{1+\varepsilon} \leq \frac{\mathrm{wd}'(v,s_v)}{1+\varepsilon} \leq \mathrm{wd}(v,s_v)$$

by (ii) and hence $\mathrm{wd}(v,w) < \mathrm{wd}(v,s_v)$. Thus we can apply (i) to obtain (iii). □

In order to show the stretch bound, we use a strategy similar to the one employed in Sect. 7, for a single level, where $s_w$ is replaced by $s'_w$. Moreover, we need to bound the additional stretch incurred by (i) using approximate source detection (costing factor $1+\mathcal{O}(\varepsilon)$) and (ii) approximating distances between skeleton nodes using the spanner $G_{\mathcal{S}}$ (costing factor $2k-1$).

**Lemma 8.4** *The above algorithm guarantees w.h.p. stretch* $(1+\mathcal{O}(\varepsilon))(4k-1)$.

***Proof*** Assume that $v \in V$ is given label $\lambda_w$ for some $w \in V$. If we route using $L_v(V)$, then $\mathrm{wd}'(v,w) \leq \mathrm{wd}'(w,s'_w) \leq \mathrm{wd}'(w,s_w)$ and Statement (ii) of Corollary 8.3 bounds the stretch by $1+\varepsilon$. Moreover, if $\mathrm{wd}(v,w) < \mathrm{wd}'(w,s'_w)/(1+\varepsilon)$, Statement (iii) of Corollary 8.3, applied to $w$, implies that

$$\begin{aligned}\mathrm{wd}'(w,v) &\leq (1+\varepsilon)\mathrm{wd}_h(w,v)\\ &= (1+\varepsilon)\mathrm{wd}(v,w)\\ &< \mathrm{wd}'(w,s'_w)\end{aligned}$$

and we route using $L_v(V)$.

Therefore, if we route via $s'_w$, it must hold that $\mathrm{wd}'(w,s'_w) \leq (1+\varepsilon)\mathrm{wd}(v,w)$ and thus also

$$\mathrm{wd}(v,s'_w) \leq \mathrm{wd}(v,w) + \mathrm{wd}(w,s'_w) \leq (2+\varepsilon)\mathrm{wd}(v,w)\,.$$

Consider a shortest path $p \in \mathrm{paths}(v,s'_w)$, i.e., $W(p) = \mathrm{wd}(v,s'_w)$ and denote by $s$ the first node on the path that is in $\mathcal{S}$. Analogously to Lemma 8.2, w.h.p. $s$ is among the first $h$ nodes of the path and hence $\mathrm{wd}_h(v,s) = \mathrm{wd}(v,s)$. From Statement (iii) of Corollary 6.12 and the above bounds, we infer that

$$\begin{aligned}&\mathrm{wd}'(v,s) + \mathrm{wd}_{\mathcal{S}}(s,s'_w) + \mathrm{wd}'(w,s'_w)\\ &\leq (1+\varepsilon)\mathrm{wd}_h(v,s) + (1+\varepsilon)(2k-1)\mathrm{wd}(s,s'_w)\\ &\quad + (1+\varepsilon)\mathrm{wd}(v,w)\\ &= (1+\varepsilon)\mathrm{wd}(v,s) + (1+\varepsilon)(2k-1)\mathrm{wd}(s,s'_w)\\ &\quad + (1+\varepsilon)\mathrm{wd}(v,w)\\ &\leq (1+\varepsilon)(2k-1)\mathrm{wd}(v,s'_w) + (1+\varepsilon)\mathrm{wd}(v,w)\\ &\leq (1+\varepsilon)(2+\varepsilon)(2k-1)\mathrm{wd}(v,w) + (1+\varepsilon)\mathrm{wd}(v,w)\\ &\in (1+\mathcal{O}(\varepsilon))(4k-1)\mathrm{wd}(v,w)\,,\end{aligned}$$

where in the last step we used the assumption that $\varepsilon \in \mathcal{O}(1)$. It follows that

$$\begin{aligned}&\min_{s\in\mathcal{S}}\{\mathrm{wd}'(v,s) + \mathrm{wd}_{\mathcal{S}}(s,s'_w) + \mathrm{wd}'(w,s'_w)\}\\ &\leq (1+\mathcal{O}(\varepsilon))(4k-1)\mathrm{wd}(v,w)\,,\end{aligned}$$

proving the stated bound on the stretch also in the second case. □

### 8.2.3 From stateful to stateless routing

From a high-level point of view, the routing is essentially already stateless: suppose a destination label $\lambda(w)$ of node $w$ given at node $v$. If we consider the "path" on node set $\mathcal{S}\cup\{v,w\}$ induced by contracting all edges of the routing path from $v$ to $w$ containing nodes $u \in V \setminus (\mathcal{S}\cup\{v,w\})$, then we route on each resulting edge $\{s,t\}$ at cost $\mathrm{wd}'(s,t)$. The main issue is that when routing over an edge $e = \{x,y\}$ towards some node $t \in \mathcal{S}\cup\{w\}$ in $G$, it is neither guaranteed that $\mathrm{wd}'(y,t) = \mathrm{wd}'(x,t) - W(e)$ nor that $(\mathrm{wd}'(x,t),t) \in L_x(\cdot)$.

To resolve this, we adapt the approach used in the remark in Sect. 5.2. For each spanner edge $e = \{s,t\} \in E_{\mathcal{S}}$, a

routing path $p(e)$ is found using Corollary 6.13, so that each $v \in p(e)$ knows the next routing hop on $p(e) = (s, \dots, v, \dots, t)$ to both $s$ and $t$, as well as the respective weights of the subpaths $W_{p(e)}(v, s) := W((s, \dots, v))$ and $W_{p(e)}(v, t) := W((v, \dots, t))$. Note that $W(p(e)) = W_{p(e)}(v, s) + W_{p(e)}(v, t) \leq W_{\mathcal{S}}(e)$. Let us extend the domain of the distance metric $\mathrm{wd}_{\mathcal{S}}$ from $\mathcal{S} \times \mathcal{S}$ to $V \times \mathcal{S}$ by setting

$$\begin{aligned}\mathrm{wd}_{\mathcal{S}}(v, s) := \min(\{\mathrm{wd}'(v, t) + \mathrm{wd}_{\mathcal{S}}(t, s) \mid t \in \mathcal{S}\} \cup \\ \{W_{p(e)}(v, t) + \mathrm{wd}_{\mathcal{S}}(s, t) \mid \\ e = \{t, t'\} \in E_{\mathcal{S}} \wedge v \in p(e)\}) .\end{aligned}$$

Intuitively, $\mathrm{wd}_{\mathcal{S}}(v, s)$ is an upper bound on the cost from routing from $v$ to $s$ based on the information from the first two steps of the algorithm, where we account for the possibility that an edge of the spanner has been "partially" traversed by following a prefix of some path $p(e)$ up to node $v$.

Let $L_{v,i}(V)$ denote the lists computed by the unweighted source detection algorithm for weight class $i$ in Step (3). To determine its distance estimate and the next routing hop to node $w$, node $v$ finds the following minimum, where $i$ ranges over all $\mathcal{O}(\varepsilon^{-1} \log n)$ weight classes used by the approximate source detection algorithm (cf. Sect. 5):

$$\min \left\{ \mathrm{wd}_{\mathcal{S}}(v, s'_w) + \mathrm{wd}'(s'_w, w), \min_{0 \leq i \leq i_{\max}} \left\{ \mathrm{wd}_i(v, w) \mid \exists (\mathrm{hd}_i(v, w), w) \in L_{v,i}(V) \right\} \right\} .$$

The next node on the routing path is then selected in accordance with the minimum. In case of a tie we give precedence to $L_{v,i}(V)$ for minimal $i$.

By construction, the modified routing scheme satisfies the following properties: (i) If $v$ computes distance estimate $\tilde{\mathrm{wd}}(v, w)$ and routes via neighbor $u$, then $u$ computes distance estimate $\tilde{\mathrm{wd}}(u, w) \leq \tilde{\mathrm{wd}}(v, w) - W(\{v, u\})$, and (ii) $\tilde{\mathrm{wd}}(v, w)$ is bounded from above by the distance estimate used in the stateful variant of the algorithm. These two properties immediately imply that the stretch guarantee of the stateful scheme carries over to the stateless scheme, and we obtain the following theorem.

**Theorem 8.5** *Given an integer $k \in [1, \log n]$ and $0 < \varepsilon \in \mathcal{O}(1)$, tables and $\mathcal{O}(\log n)$-bit labels for routing and distance approximation with stretch $(1 + \mathcal{O}(\varepsilon))(4k - 1)$ can be computed in $\tilde{\mathcal{O}}(\varepsilon^{-2} n^{\frac{2k+1}{4k}} + D)$ rounds w.h.p.*

## 9 Lower bounds

In this section we prove that the asymptotic complexity of our algorithms is nearly the best possible within the **CONGEST** model. We start with a lower bound on the time required to estimate the diameter of the network, which is immediately applicable to, say, APSP distance estimation.

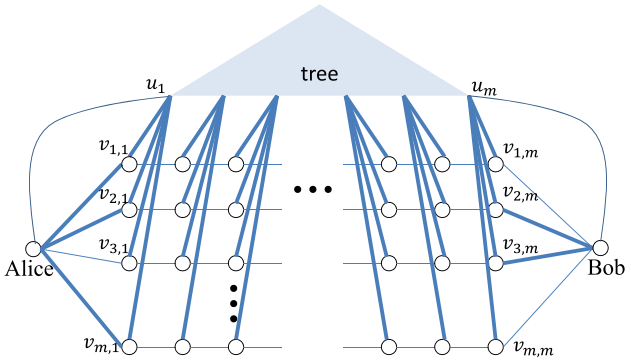


**Fig. 5** An illustration of the graph used in the proof of Theorem 9.1. Thick edges denote edges of weight $\omega_{\max}$, other edges are of weight 1. The shaded triangle represents a binary tree

### 9.1 Approximating the diameter in weighted graphs

Frischknecht et al. [19] show that approximating the diameter of an unweighted graph to within a factor smaller than 1.5 cannot be done in the **CONGEST** model in $o(n/\log n)$ time. Here, following the framework of Das Sarma et al. [15], we prove a hardness result for the weighted diameter, formally stated as follows.

**Theorem 9.1** *For any $\omega_{\max} \geq \sqrt{n}$, there is a function $\alpha(n) \in \Omega(\omega_{\max}/\sqrt{n})$ such that the following holds. In the family of weighted graphs of hop-diameter $D \in \mathcal{O}(\log n)$ and edge weights 1 and $\omega_{\max}$ only, an (expected) $\alpha(n)$-approximation of the weighted diameter requires $\tilde{\Omega}(\sqrt{n})$ communication rounds in the **CONGEST** model.*

***Proof*** Let $n \in \mathbb{N}$. Like in [15], we construct a graph family $\mathcal{G}_n$ where each $G \in \mathcal{G}_n$ has $\Theta(n)$ nodes. Let $m = \lceil \sqrt{n} \rceil$. All graphs in $\mathcal{G}_n$ consist of the following three conceptual parts. Figure 5 illustrates a part of the construction.

- Nodes $v_{i,j}$ for $1 \leq i, j \leq m$. These nodes are connected as $m$ paths of length $m - 1$ (horizontal paths in the figure). All path edges are of weight 1.
- A star rooted at an *Alice* node, whose the children are $v_{1,1}, \dots, v_{m,1}$, and similarly, a star rooted at a *Bob* node, whose leaves are $v_{1,m}, \dots, v_{m,m}$. The weights of these edges may be either 1 or $\omega_{\max}$ (that's the only difference between graphs in $\mathcal{G}_n$).
- For each $1 \leq j \leq m$ there is a node $u_j$ connected to all nodes $v_{i,j}$, $1 \leq i \leq m$ in "column" $j$, with edges of weight $\omega_{\max}$. In addition, there is a binary tree whose leaves are the nodes $u_j$. All tree edges have weight 1. Finally, *Alice* and *Bob* are connected to $u_1$ and $u_m$, respectively, by edges of weight 1.

Clearly, the hop-diameter of any graph in $\mathcal{G}_n$ is $\mathcal{O}(\log n)$: the hop-distance from any node to one of the nodes $u_j$ is $\mathcal{O}(\log n)$, and the distance between any two such nodes is also $\mathcal{O}(\log n)$. Furthermore, the following fact is shown by Das Sarma et al. [15], based on the two-party communication complexity of deciding *set disjointness*.

**Fact 9.1** *(Complexity of Set Disjointness [15]) Let* $\mathcal{M} = \{1, \ldots, m\}$. *Suppose that* Alice *holds a set* $A \subseteq \mathcal{M}$ *and that* Bob *holds a set* $B \subseteq \mathcal{M}$. *If deciding whether* $A \cap B = \emptyset$ *can be reduced to running a* ***CONGEST*** *algorithm on* $\mathcal{G}_n$ *(where edge weights incident to the* Alice *node depend only on* $A$ *and those incident to the* Bob *node depend only on* $B$*), then this algorithm runs for* $\tilde{\Omega}(m)$ *rounds, even if it is randomized.*

Accordingly, we now show that if the diameter of $G \in \mathcal{G}_n$ can be approximated within factor $\omega_{\max}/\sqrt{n}$ in time $T$ in the **CONGEST** model, then set disjointness can be decided in time $T + 1$. To this end, we set the edge weights of the stars rooted at *Alice* and *Bob* as follows: for all $i \in \{1, \ldots, m\}$, the edge from *Alice* to $v_{i,1}$ has weight $\omega_{\max}$ if $i \in A$ and weight 1 else; likewise, the edge from *Bob* to $v_{i,m}$ has weight $\omega_{\max}$ if $i \in B$ and weight 1 otherwise.

Note that given $A$ at *Alice* and $B$ at *Bob*, we can inform the nodes $v_{i,1}$ and $v_{i,m}$ of these weights in one round. Now run any algorithm that outputs a value between WD (the weighted diameter) and $\alpha(n)\mathrm{WD} := \omega_{\max}\mathrm{WD}/(\sqrt{n} + C \log n)$ (for a suitable constant $C$) within $T$ rounds, and output "$A$ and $B$ are disjoint" if the outcome is at most $\omega_{\max}$ and output "$A$ and $B$ are not disjoint" othwerwise.

It remains to show that the outcome of this computation is correct for any inputs $A$ and $B$ and the statement of the theorem will follow from Fact 9.1 (recall that the number of nodes in $G$ is $\Theta(n)$). Suppose first that $A \cap B = \emptyset$. Then for each node $v_{i,j}$, there is a path of at most $\sqrt{n}$ edges of weight 1 connecting it to *Alice* or *Bob*, and *Alice* and *Bob* are connected to all nodes in the binary tree and each other via $\mathcal{O}(\log n)$ hops in the binary tree (whose edges have weight 1 as well). Hence the weighted diameter of $G$ is $\sqrt{n}+\mathcal{O}(\log n)$ in this case and the output is correct (where we assume that $C$ is sufficiently large to account for the $\mathcal{O}(\log n)$ term). Now suppose that $i \in A \cap B$. In this case each path from node $v_{i,1}$ to *Bob* contains an edge of weight $\omega_{\max}$, since the edges from *Alice* to $v_{i,1}$ and *Bob* to $v_{i,m}$ as well as those connecting $v_{i,j}$ to $u_j$ have weight $\omega_{\max}$. Hence, the weighted distance from $v_{i,1}$ to *Bob* is strictly larger than $\omega_{\max}$ and the output is correct as well. This shows that set disjointness is decided correctly and therefore the proof is complete. □

## 9.2 Hardness of name-dependent distributed table construction

A lower bound on name-dependent distance approximation follows directly from Theorem 9.1.

**Corollary 9.2** *For any* $\omega_{\max} \geq \sqrt{n}$, *there is a function* $\alpha(n) \in \Omega(\omega_{\max}/\sqrt{n})$ *such that the following holds. In the family of weighted graphs of hop-diameter* $D \in \mathcal{O}(\log n)$ *and edge weights 1 and* $\omega_{\max}$ *only, constructing labels of size* $\tilde{o}(\sqrt{n})$ *and tables for distance approximation of (expected) stretch* $\alpha(n)$ *requires* $\tilde{\Omega}(\sqrt{n})$ *communication rounds in the* ***CONGEST*** *model.*

***Proof*** We use the same construction as in the previous proof, however, now we need to solve the disjointness problem using the tables and lables. Using the same setup, we run the assumed table and label construction algorithm. Afterwards, we transmit, e.g., the label of *Alice* to all nodes $v_{i,1}$. This takes $\tilde{o}(\sqrt{n})$ rounds by assumption on label size. We then query the estimated distance to *Alice* at the nodes $v_{i,1}$ and collect the results at *Alice*. Analogously to the proof of Theorem 9.1, the maximum of these values is large if and only if the input satisfies that $A \cap B = \emptyset$. Since transmitting the label costs only $\tilde{o}(\sqrt{n})$ additional rounds, the same asymptotic lower bound as in Theorem 9.1 follows. □

A variation of the theme shows that stateless routing requires $\tilde{\Omega}(\sqrt{n})$ time.

**Corollary 9.3** *For any* $\omega_{\max} \geq \sqrt{n}$, *there is a function* $\alpha(n) \in \Omega(\omega_{\max}/\sqrt{n})$ *such that the following holds. In the family of weighted graphs of hop-diameter* $D \in \mathcal{O}(\log n)$ *and edge weights 1 and* $\omega_{\max}$ *only, constructing stateless routing tables of (expected) stretch* $\alpha(n)$ *with labels of size* $\tilde{o}(\sqrt{n})$ *requires* $\tilde{\Omega}(\sqrt{n})$ *communication rounds in the* ***CONGEST*** *model.*

***Proof*** We consider the graph $G_n$ as defined in the proof of Theorem 9.1 and input sets $A$ and $B$ at *Alice* and *Bob*, respectively, but we use a different assignment of edge weights.

- All edges incident to a node in the binary tree have weight $\omega_{\max}$.
- For each $i \in \{1, \ldots, m\}$, the edge from *Alice* to $v_{i,1}$ has weight 1 if $i \in A$ and weight $\omega_{\max}$ else. Likewise, the edge from *Bob* to $v_{i,m}$ has weight 1 if $i \in B$ and otherwise weight $\omega_{\max}$.
- The remaining edges (on the $m$ paths from $v_{i,1}$ to $v_{i,m}$) have weight 1.

Observe that the distance from *Alice* to *Bob* is $\sqrt{n} + 1$ if $A \cap B \neq \emptyset$ and at least $\omega_{\max} + 2$ if $A \cap B = \emptyset$. Once static routing tables for routing on paths of stretch at most $\omega_{\max}/(\sqrt{n} + 1)$ are set up, e.g. *Bob* can decide whether $A$ and $B$ are disjoint as follows. *Bob* sends its label to *Alice*

via the binary tree (which takes time $\tilde{o}(\sqrt{n})$ if the label has size $\tilde{o}(\sqrt{n})$). *Alice* responds with "$i$" if the first routing hop from *Alice* to *Bob* is node $v_{i,1}$ and $i \in A$ (i.e., the weight of the edge is 1), and "$A \cap B = \emptyset$" else (this takes $\mathcal{O}(\log n)$ rounds). *Bob* then outputs "$A \cap B \neq \emptyset$" if *Alice* responded with "$i$" and $i \in B$ (i.e., the weight of the routing path is $\sqrt{n}+1$ since the edge from *Bob* to $v_{i,m}$ has weight 1) and "$A \cap B = \emptyset$" otherwise.

If the output is "$A \cap B \neq \emptyset$", it is correct because $i \in A \cap B$. On the other hand, if it is "$A \cap B = \emptyset$", the route from *Alice* to *Bob* must contain an edge of weight $\omega_{\max}$, implying by the stretch guarantee that there is no path of weight $\sqrt{n}+1$ from *Alice* to *Bob*. This in turn entails that $A \cap B = \emptyset$ due to the assignment of weights and we conclude that the output is correct also in this case. Hence the statement of the corollary follows from Fact 9.1. □

As a final remark, we point out that name-independent routing (i.e., $\lambda(v) = v$ for all $v \in V$) requires $\tilde{\Omega}(n)$ rounds, which is shown by similar techniques [32,39]. Thus, relabeling is essential for achieving small running times.

**Acknowledgements** Open access funding provided by Max Planck Society.



## References

1. Abboud, A., Censor-Hillel, K., Khoury, S.: Near-linear lower bounds for distributed distance computations, even in sparse networks. In: Proceedings of the International Symposium on Distributed Computing (DISC), pp. 29–42 (2016). https://doi.org/10.1007/978-3-662-53426-7_3
2. Antonio, J., Huang, G., Tsai, W.: A fast distributed shortest path algorithm for a class of hierarchically clustered data networks. IEEE Trans. Comput. **41**, 710–724 (1992)
3. Awerbuch, B., Bar-Noy, A., Linial, N., Peleg, D.: Compact distributed data structures for adaptive network routing. In: Proceedings of the 21st ACM Symposium on Theory of Computing, pp. 230–240 (1989)
4. Awerbuch, B., Bar-Noy, A., Linial, N., Peleg, D.: Improved routing strategies with succinct tables. J. Algorithms **11**(3), 307–341 (1990)
5. Awerbuch, B., Berger, B., Cowen, L., Peleg, D.: Near-linear cost sequential and distributed constructions of sparse neighborhood covers. In: Proceedings 34th Symposium on Foundations of Computer Science (FOCS), pp. 638–647 (1993)
6. Awerbuch, B., Peleg, D.: Routing with polynomial communication-space trade-off. SIAM J. Discr. Math. **5**, 151–162 (1992)
7. Baswana, S., Kavitha, T.: Faster algorithms for approximate distance oracles and all-pairs small stretch paths. In: Proceedings of the 47th Symposium on Foundations of Computer Science (FOCS), pp. 591–602 (2006)
8. Baswana, S., Sen, S.: Approximate distance oracles for unweighted graphs in expected $O(n^2)$ time. ACM Trans. Algorithms **2**, 557–577 (2006)
9. Baswana, S., Sen, S.: A simple and linear time randomized algorithm for computing sparse spanners in weighted graphs. Random Struct. Algorithms **30**(4), 532–563 (2007)
10. Becker, R., Karrenbauer, A., Krinninger, S., Lenzen, C.: Near-optimal approximate shortest paths and transshipment in distributed and streaming models. In: 31st Symposium on Distributed Computing (DISC) (2017)
11. Bellman, R.E.: On a routing problem. Quart. Appl. Math. **16**, 87–90 (1958)
12. Bernstein, A.: Maintaining shortest paths under deletions in weighted directed graphs: [extended abstract]. In: Proceedings of the 45th Symposium Theory of Computing (STOC), pp. 725–734 (2013)
13. Cicerone, S., D'Angelo, G., Di Stefano, G., Frigioni, D., Petricola, A.: Partially dynamic algorithms for distributed shortest paths and their experimental evaluation. J. Comput. **2**, 16–26 (2007)
14. Das Sarma, A., Dinitz, M., Pandurangan, G.: Efficient computation of distance sketches in distributed networks. In: Proceedings of the 24th ACM Symposium on Parallelism in Algorithms and Architectures (2012)
15. Das Sarma, A., Holzer, S., Kor, L., Korman, A., Nanongkai, D., Pandurangan, G., Peleg, D., Wattenhofer, R.: Distributed verification and hardness of distributed approximation. In: Proceedings of the 43th ACM Symposium on Theory of Computing, pp. 363–372 (2011)
16. Derbel, B., Gavoille, C., Peleg, D., Viennot, L.: On the locality of distributed sparse spanner construction. In: Proceedings of the 27th Symposium on Principles of Distributed Computing (PODC), pp. 273–282 (2008)
17. Elkin, M., Neiman, O.: On efficient distributed construction of near optimal routing schemes: Extended abstract. In: Proceedings 34th Symposium on Principles of Distributed Computing (PODC), pp. 235–244 (2016)
18. Ford, L.R.: Network flow theory. Techical Report P-923, The Rand Corporation (1956)
19. Frischknecht, S., Holzer, S., Wattenhofer, R.: Networks cannot compute their diameter in sublinear time. In: Proceedings of the 23rd ACM-SIAM Symposium on Discrete Algorithms, pp. 1150–1162 (2012)
20. Gavoille, C., Peleg, D.: Compact and localized distributed data structures. Distrib. Comput. **16**, 111–120 (2003)
21. Gavoille, C., Peleg, D., Pérennes, S., Raz, R.: Distance labeling in graphs. In: Proceedings of the 12th ACM Symposium on Discrete Algorithms, pp. 210–219 (2001)
22. Ghaffari, M., Lenzen, C.: Near-optimal distributed tree embedding. In: 28th Symposium on Distributed Computing (DISC), pp. 197–211 (2014)
23. Haldar, S.: An 'all pairs shortest paths' distributed algorithm using $2n^2$ messages. J. Algorithms **24**(1), 20–36 (1997)
24. Henzinger, M., Krinninger, S., Nanongkai, D.: An Almost-Tight Distributed Algorithm for Computing Single-Source Shortest Paths. CoRR **abs/1504.07056** (2015)
25. Holzer, S., Pinsker, N.: Approximation of Distances and Shortest Paths in the Broadcast Congest Clique. CoRR **abs/1412.3445** (2014)
26. Holzer, S., Wattenhofer, R.: Optimal distributed all pairs shortest paths and applications. In: Proceedings of the 31st ACM Symposium on Principles of Distributed Computing (2012)
27. Hua, Q.S., Fan, H., Qian, L., Ai, M., Li, Y., Shi, X., Jin, H.: Brief Announcement: A Tight Distributed Algorithm for All Pairs Shortest Paths and Applications. In: 28th ACM Symposium on Parallelism in Algorithms and Architectures, pp. 439–441 (2016)

28. Izumi, T., Wattenhofer, R.: Time lower bounds for distributed distance oracles. In: Proceedings of the 18th International Conference on Principles of Distributed Systems (OPODIS), pp. 60–75. Springer International Publishing (2014). https://doi.org/10.1007/978-3-319-14472-6_5
29. Kanchi, S., Vineyard, D.: An optimal distributed algorithm for all-pairs shortest-path. Int. J. Inf. Theories Appl. **11**(2), 141–146 (2004)
30. Kavitha, T.: Faster algorithms for all-pairs small stretch distances in weighted graphs. In: Proceedings of the FSTTCS, pp. 328–339 (2007)
31. Klein, P.N., Subramanian, S.: A fully dynamic approximation scheme for shortest paths in planar graphs. Algorithmica **22**, 235–249 (1998)
32. Lenzen, C., Patt-Shamir, B.: Fast routing table construction using small messages [Extended Abstract]. In: Proceedings of the 45th Symposium on the Theory of Computing (STOC) (2013). Full version at http://arxiv.org/abs/1210.5774
33. Lenzen, C., Patt-Shamir, B.: Improved distributed steiner forest construction. In: Proceedings of the 32nd Symposium on Principles of Distributed Computing (PODC), pp. 262–271 (2014)
34. Lenzen, C., Patt-Shamir, B.: Fast partial distance estimation and applications. In: Proceedings of the 33rd Symposium on Principles of Distributed Computing (PODC) (2015)
35. Lenzen, C., Peleg, D.: Efficient distributed source detection with limited bandwidth. In: Proceedings of the 32nd ACM Symposium on Principles of Distributed Computing (2013)
36. Madry, A.: Faster approximation schemes for fractional multicommodity flow problems via dynamic graph algorithms. In: Proceedings of the 42nd ACM Symposium on Theory of Computing, STOC 2010, Cambridge, Massachusetts, USA, 5–8 June 2010, pp. 121–130 (2010)
37. McQuillan, J., Richer, I., Rosen, E.: The new routing algorithm for the arpanet. IEEE Trans. Commun. **COM–28**(5), 711–719 (1980)
38. Moy, J.: OSPF version 2. RFC 2328, Network Working Group (1998)
39. Nanongkai, D.: Distributed approximation Algorithms for weighted shortest paths. In: Proceedings of the 46th Symposium on Theory of Computing (STOC), pp. 565–573 (2014)
40. Peleg, D.: Proximity-preserving labeling schemes and their applications. In: Proceedings of the 25th International Workshop on Graph-Theoretic Concepts in Computer Science, pp. 30–41 (1999)
41. Peleg, D.: Distributed Computing: A Locality-Sensitive Approach. SIAM, Philadelphia, PA (2000)
42. Peleg, D., Roditty, L., Tal, E.: Distributed algorithms for network diameter and girth. In: Proceedings of the 39th International Colloquium on Automata, Languages, and Programming (2012)
43. Peleg, D., Rubinovich, V.: A near-tight lower bound on the time complexity of distributed minimum-weight spanning tree construction. SIAM J. Comput. **30**, 1427–1442 (2000)
44. Peleg, D., Schäffer, A.A.: Graph spanners. J. Graph Theory **13**(1), 99–116 (1989)
45. Peleg, D., Ullman, J.D.: An optimal synchronizer for the hypercube. SIAM J. Comput. **18**(2), 740–747 (1989)
46. Peleg, D., Upfal, E.: A trade-off between space and efficiency for routing tables. J. ACM **36**(3), 510–530 (1989)
47. Peterson, L.L., Davie, B.S.: Computer Networks: A Systems Approach, 5th edn. Morgan Kaufmann, Burlington (2011)
48. Raghavan, P., Thompson, C.D.: Provably good routing in graphs: Regular arrays. In: Proceedings of the 17th Annual ACM Symposium on Theory of Computing, STOC '85, pp. 79–87 (1985)
49. Roditty, L., Thorup, M., Zwick, U.: Deterministic constructions of approximate distance oracles and spanners. In: Proceedings of the 32nd Colloquium on Automata, Languages, and Programming (ICALP), pp. 261–272 (2005)
50. Santoro, N., Khatib, R.: Labelling and implicit routing in networks. Comput. J. **28**, 5–8 (1985)
51. Segall, A.: Distributed network protocols. IEEE Trans. Inf. Theory **29**, 23–35 (1983)
52. Thorup, M., Zwick, U.: Compact routing schemes. In: Proceedings of the 13th ACM Symposium on Parallel Algorithms and Architectures (2001)
53. Thorup, M., Zwick, U.: Approximate distance oracles. J. ACM **52**(1), 1–24 (2005)