Roundtrip Spanners and Roundtrip Routing in Directed Graphs

IAM RODITTY

Weizmann Institute

MIKKEL THORUP

AT&T Labs – Research

AND

URI ZWICK

Tel Aviv University

29

Abstract. We introduce the notion of *roundtrip-spanners* of weighted *directed* graphs and describe efficient algorithms for their construction. We show that for every integer $k \geq 1$ and any $\epsilon > 0$, any directed graph on n vertices with edge weights in the range $[1, W]$ has a $(2k + \epsilon)$-roundtrip-spanner with $O(\min\{(k^2/\epsilon)\, n^{1+1/k}\, \log(nW),\ (k/\epsilon)^2\, n^{1+1/k}\, (\log n)^{2-1/k}\})$ edges. We then extend these constructions and obtain compact roundtrip routing schemes. For every integer $k \geq 1$ and every $\epsilon > 0$, we describe a roundtrip routing scheme that has stretch $4k + \epsilon$, and uses at each vertex a routing table of size $\tilde{O}((k^2/\epsilon)n^{1/k} \log(nW))$. We also show that any weighted directed graph with *arbitrary* positive edge weights has a 3-roundtrip-spanner with $O(n^{3/2})$ edges. This result is optimal. Finally, we present a stretch 3 roundtrip routing scheme that uses local routing tables of size $\tilde{O}(n^{1/2})$. This routing scheme is essentially optimal. The roundtrip-spanner constructions and the roundtrip routing schemes for directed graphs that we describe are only slightly worse than the best available spanners and routing schemes for undirected graphs. Our roundtrip routing schemes substantially improve previous results of Cowen and Wagner. Our results are obtained by combining ideas of Cohen, Cowen and Wagner, Thorup and Zwick, with some new ideas.

Categories and Subject Descriptors: F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems—*Computations on discrete structures*; G.2.2 [**Discrete Mathematics**]: Graph Theory—*Graph algorithms*

A preliminary version of this article appeared in *Proceedings of the 13th Annual Symposium on Discrete Algorithms* (*SODA'02*), ACM, New York, 2002, pp. 844–851.

The research of M. Thorup and U. Zwick was supported by BFS grant no. 2006261.

Authors' addresses: L. Roditty, Faculty of Mathematics and Computer Science, Weizmann Institute, Rehovot 76100, Israel, e-mail: liam.roditty@weizmann.ac.il; M. Thorup, AT&T Labs – Research, 180 Park Avenue, Florham Park, NJ 07932, e-mail: mthorup@research.att.com; U. Zwick, School of Computer Science, Tel-Aviv University, Tel-Aviv 69978, Israel, e-mail: zwick@tau.ac.il.

General Terms: Algorithms, Theory

Additional Key Words and Phrases: Routing, spanners, roundtrip, distances, shortest paths

ACM Reference Format:

Roditty, L., Thorup, M., and Zwick, U. 2008. Roundtrip spanners and roundtrip Routing in directed graphs. ACM Trans. Algor. 4, 3, Article 29 (June 2008), 17 pages. DOI = 10.1145/1367064.1367069 http://doi.acm.org/10.1145/1367064.1367069

1. *Introduction*

The main results of this article are improved roundtrip routing schemes for directed graphs. To obtain a clearer presentation of these schemes, and the techniques used to construct them, we begin with a discussion of spanners and roundtrip-spanners.

A spanning tree is the sparsest subgraph that preserves the connectivity of a graph. A *spanner* is a relatively sparse subgraph that preserves, to a certain extent, the *distances* in a graph. More formally, let $G = (V, E)$ be an undirected graph with a weight (or length) function $w : E \to \mathbb{R}^+$ defined on its edges. If $u, v \in V$, we let $\delta_G(u, v)$ be the distance between u and v in G. A subgraph $H = (V, E')$ of G is said to be a *t-spanner* of G if and only if $\delta_H(u, v) \leq t \cdot \delta_G(u, v)$, for every $u, v \in V$. The number t is said to be the *stretch factor* of the spanner. For any integer $k \geq 1$, any weighted undirected graph on n vertices has a $(2k - 1)$-spanner with $O(n^{1+1/k})$ edges. (See Althöfer et al. [1993], Peleg [2000], Thorup and Zwick [2005].) This trade-off between the *stretch* of a spanner and its size is known to be tight for $k = 1, 2, 3, 5$, and is believed to be tight for any k. (See, e.g., the discussion in Thorup and Zwick [2005].) Spanners have many applications in distributed computing (see Peleg [2000]).

Can the notion of spanners be extended to directed graphs? The definition of spanners still makes sense in the directed setting, but the notion becomes uninteresting. Take a complete directed bipartite graph. Removing any edge changes the reachability in the graph, so no strict subgraph is a t-spanner of this graph, for any finite t. Thus, no general tradeoff between stretch and size is possible for directed graphs.

Cowen and Wagner [1999, 2000], working on routing problems, made the very nice observation that some of the techniques related to approximated distances and paths in undirected graphs can be used on directed graphs, if *roundtrip distances*, instead of *one way* distances are considered. Here the roundtrip distance from u to v is the one-way distance from u to v plus the one way distance from v to u. Roundtrip distances in directed graphs are very natural generalizations of distances in undirected graphs. In particular, roundtrip distances are symmetric, and hence they define a metric for directed graphs.

Implicit in the work of Cowen and Wagner [1999, 2000] is the notion of *roundtrip spanners*. We make this notion explicit and obtain vastly improved constructions of roundtrip spanners. Let $G = (V, E)$ be a *directed* graph with a weight (or length) function $w : E \to \mathbb{R}^+$ defined on its edges. A subgraph $H = (V, E')$ of G is said to be a *t-roundtrip-spanner* of G if and only if $\delta_H(u \rightleftarrows v) \leq t \cdot \delta_G(u \rightleftarrows v)$, for every $u, v \in V$. (Here $\delta_G(u \rightleftarrows v) = \delta_G(u \to v) + \delta_G(v \to u)$ is the roundtrip distance between u and v in G.)

The results of Cowen and Wagner [1999, 2000] imply that, for any integer $k \geq 1$, any weighted directed graph on n vertices has a $(2^k - 1)$-roundtrip-spanner of size

$\tilde{O}(n^{1+1/k})$. For $k = 2$, this result is very close to being optimal. For larger values of k, the stretch obtained is extremely large. We show that for any $k \geq 1$ and any fixed $\epsilon > 0$, any weighted directed graph on n vertices with edge weights in the range $[1, W]$ has a $(2k + \epsilon)$-roundtrip-spanner of size $O((k^2/\epsilon)\, n^{1+1/k} \log(nW))$. We can also remove the dependence of the number of edges in the spanner on W, and show that any directed graph on n vertices with arbitrary positive edge weights has a $(2k + \epsilon)$-roundtrip-spanner of size $O((k/\epsilon)^2\, n^{1+1/k} (\log n)^{2-1/k})$. These results are very close to the best available results for undirected graphs cited above. For $k = 2$ we have a specialized construction that shows that any n-vertex digraph has a 3-roundtrip-spanner of size $O(n^{3/2})$. This improves the corresponding result of Cowen and Wagner [1999, 2000] by a polylogarithmic factor and makes it completely tight.

There are several different techniques for constructing sparse spanners for undirected graphs. (See Althöfer et al. [1993], Awerbuch et al. [1999], Cohen [1999], Peleg [2000], and Thorup and Zwick [2005].) Interestingly, some of these techniques can be adapted to the directed setting while others cannot, at least not in an obvious way. The technique of Thorup and Zwick [2005], for example, which gives the fastest way of constructing $(2k - 1)$-spanners of size $O(n^{1+1/k})$ for undirected graphs, extends to the directed case when $k = 2$, but not for larger values of k. Using it we obtain the optimal construction of 3-roundtrip-spanners. To obtain sparser roundtrip-spanners with a larger stretch, we use an extension of the nice technique of Cohen [1999].

One of the uses of spanners in distributed computing is the design of compact *routing schemes*. Let $G = (V, E)$ be a graph with *port numbers* assigned to its links. (In a directed graph a link is simply a directed edge. In undirected graphs, each undirected edge represent two links, one in each direction.) The port number assigned to each edge is unique. A routing scheme assigns to each vertex $v \in V$ a short label, or name, denoted $label(v)$. Each vertex $v \in V$ also has a local routing table $RT(v)$. A vertex u that wants to send a packet p to vertex v is assumed to know $label(v)$. It constructs a message $(label(v), p)$ that contains $label(v)$ as its *header* and passes it to its routing handler. When a vertex w of the network receives a message with $label(v)$ as its header, it has to decide, based on $label(v)$ and on its local routing table $RT(w)$, on which out-going link to forward this message. This out-going link is identified by its port number. For more on routing schemes, see Peleg [2000] and Thorup and Zwick [2001].

The *stretch* of a routing scheme is the worst possible ratio between the length of the route that a message follows from u to v, divided by the distance from u to v in the graph. The *roundtrip-stretch* of a routing scheme is the corresponding quantity for a roundtrip route from u to v and back.

It is easy to obtain stretch 1, that is, route along shortest paths, if a full routing table of size $O(n)$ is stored at each vertex. (We measure the size of a routing table in $O(\log n)$-bit words.) However, it is unrealistic to expect each vertex of a network to hold a routing table that is almost as big as the network itself. Hence, there is great interest in what stretches are attainable with much smaller routing tables. Preferably, we would like *all* routing tables to be small.

Thorup and Zwick [2001], improving many previous results, obtained essentially optimal routing schemes for undirected graphs. For every integer $k \geq 1$, they describe a routing scheme that uses $o(\log^2 n)$-bit labels, local routing tables of size $\tilde{O}(n^{1/k})$, and has stretch $2k - 1$. For $k \geq 3$, these routing schemes rely on an initial

handshaking process (see Thorup and Zwick [2001] for details). Without such a handshaking process, the stretch increases to $4k - 5$.

Many real communication networks cannot be modeled accurately by undirected graphs. Some of the links in a network may be one-directional, and in some of the bidirectional links, the speed, or capacity, may depend on the direction. Thus, the study of routing schemes for *directed* graphs is well motivated. However, attempts to design compact routing schemes for directed graphs meet essentially the same obstacles met when trying to construct spanners for directed graphs.

It is in this context that Cowen and Wagner [1999, 2000] suggest the consideration of *roundtrip routing schemes*. This is also supported by the fact that many of the real-life applications actually require roundtrip routes. The TCP protocol, for example, requires an acknowledgment to be sent after each packet is received.

Cowen and Wagner [1999, 2000] describe, for any integer $k \geq 1$, a roundtrip routing scheme of stretch $2^k - 1$ that uses local routing tables of *average* size $\tilde{O}(n^{1/k})$. The size of some of the routing tables in their schemes may be as large as $\Omega(n)$. They also describe a variant of their scheme in which the *maximum* table size is $\tilde{O}(n^{(3^{k-1}+1)/(2\cdot 3^{k-1})})$ and the stretch is still $2^k - 1$. Note, however, that the maximum table size is still $\Omega(n^{1/2})$, for any $k \geq 1$.

We substantially improve these results and obtain, for any integer $k \geq 1$ and any fixed $\epsilon > 0$, a roundtrip routing scheme of stretch $4k + \epsilon$ and *maximum* table size $\tilde{O}(n^{1/k})$, assuming that the edge weights are polynomially bounded. With routing tables of *total* size $\tilde{O}(n^{1+1/k})$, we thus get stretch $4k + \epsilon$, instead of $2^k - 1$. Furthermore, we do that while keeping the maximum table size at $\tilde{O}(n^{1/k})$, instead of $\Omega(n^{1/2})$. For $k = 2$, we have an improved roundtrip routing scheme that has stretch 3 and uses routing tables of maximum size $\tilde{O}(n^{1/2})$. This scheme is essentially optimal. Our roundtrip routing schemes do *not* use handshaking.

Our improvements, for general k, over the results of Cowen and Wagner [1999, 2000] are based on a construction of Cohen [1999] that predates the results of Cowen and Wagner [1999, 2000]. Our improvement for stretch 3 extends a recent work by Thorup and Zwick [2005, 2001].

Note that the stretch, $4k + \epsilon$, of our roundtrip routing schemes is about twice as large as the stretch of our spanners. This is the price paid for being able to make local routing decisions. On the positive side, note that our roundtrip routing schemes do not lag much behind the best known routing schemes for undirected graphs that do not use handshaking that, for the same table size, have stretch $4k - 5$ (see [Thorup and Zwick 2001]).

The rest of this article is organized as follows: In the next section, we present efficient constructions of sparse roundtrip-spanners. In Section 4, we present our roundtrip routing schemes. In Section 5, we present an essentially optimal roundtrip routing scheme with stretch 3. We end in Section 6 with some concluding remarks and open problems.

2. *Roundtrip-Spanners*

Let $G = (V, E)$ be an undirected graph with a weight (or length) function $w : E \rightarrow \mathbb{R}^+$ defined on its edges. Recall that a subgraph $H = (V, E')$ of G is said to be a t-spanner of G if and only if $\delta_H(u, v) \leq t \cdot \delta_G(u, v)$, for every $u, v \in V$. (Here $\delta_G(u, v)$ is the distance between u and v in G.) For any integer $k \geq 1$, any weighted

undirected graph on n vertices has a $(2k-1)$-spanner with $O(n^{1+1/k})$ edges. (See Althöfer et al. [1993], Peleg [2000], and Thorup and Zwick [2005].)

The spanner definition can also be applied to directed graphs. But, as mentioned in the introduction, there are n-vertex graphs with $\Omega(n^2)$ edges that do not have nontrivial spanners. (Consider a complete bipartite graph with all edges directed in the same direction.)

Cowen and Wagner [1999, 2000], in the context of routing problems, suggested considering *roundtrip* distances in directed graphs, instead of *one-way* distances. They showed that some of the techniques that were used in connection with distances in undirected graphs could be used in connection with roundtrip distances in directed graphs.

Let $G = (V, E)$ be a weighted directed graph. We let $\delta(u \rightarrow v)$ and $\delta(v \leftarrow u)$ be the *one-way* distance from u to v in the graph G. We let $\delta(u \rightleftarrows v) = \delta(u \rightarrow v) + \delta(u \leftarrow v)$ be the *roundtrip* distance from u to v in G. Clearly, $\delta(u \rightleftarrows v) = \delta(v \rightleftarrows u)$. When we want to stress the fact that the distances considered are in the graph G, we add G as a subscript, as in $\delta_G(u \rightarrow v)$ and $\delta_G(u \rightleftarrows v)$. The following definition is now natural:

Definition 2.1 (ROUNDTRIP-SPANNERS). Let $G = (V, E)$ be a weighted directed graph. A subgraph $H = (V, E')$ is said to be a t-roundtrip-spanner of G if and only if $\delta_H(u \rightleftarrows v) \leq t \cdot \delta_G(u \rightleftarrows v)$, for every $u, v \in V$.

Note that, if $G = (V, E)$ is an undirected graph, then $\delta_G(u \rightleftarrows v) = 2\delta(u, v)$, for every $u, v \in V$, and thus roundtrip-spanners of undirected graphs are just conventional spanners.

Although the explicit definition of roundtrip-spanners is new, the following result is implicit in Cowen and Wagner [2000]:

THEOREM 2.2 (COWEN AND WAGNER 2000). *For every integer $k \geq 1$, every weighted directed graph on n vertices has a $(2^k - 1)$-roundtrip-spanner with $\tilde{O}(n^{1+1/k})$ edges.*

We substantially improve this result and prove:

THEOREM 2.3. *For every integer $k \geq 1$ and every $\epsilon > 0$, every weighted directed graph on n vertices with edge weights taken from $[1, W]$ has a $(2k + \epsilon)$-roundtrip-spanner with $O((k^2/\epsilon)\, n^{1+1/k} \log(nW))$ edges.*

If the edge weights in the graph are of polynomial size, then, for roughly the same number of edges, the stretch is reduced from $2^k - 1$ to $2k + \epsilon$, an exponential improvement. The rest of this section is devoted to the proof of Theorem 2.3, which uses an adaptation a technique introduced by Cohen [1999] for undirected graphs.

Let $G = (V, E)$ be an n-vertex graph with edge weights taken from $[1, W]$. Let $1 \leq R \leq 2nW$. Following Cohen [1999], we construct a subgraph $H = (V, E')$ of G, with $|E'| = O(kn^{1+1/k})$, such that, for every $u, v \in V$, if $\delta_G(u \rightleftarrows v) \leq R$, then $\delta_H(u \rightleftarrows v) \leq 2k \cdot R$. Clearly, if we take the union of such subgraphs, corresponding to $R = (1+\epsilon)^i$, for $1 \leq i \leq \log_{1+\epsilon}(2nW)$, we get a $2k(1+\epsilon)$-roundtrip-spanner of G with $O(kn^{1+1/k} \log_{1+\epsilon}(nW))$ edges.

The subgraph $H = H(R)$ is constructed as follows. Let $V' \subseteq V$ be a subset of vertices. For every $v \in V$ and $r \geq 0$, we let $\text{ball}_{V'}(v, r)$ be the set of vertices that are

```
algorithm cover(G, k, R)

C ← φ ; V_{k-1} ← V
for i ← k − 1 downto 0
{
    S_i ← sample(V_i, n^{−i/k})
    C ← C ∪ { ball_{V_i}(v, (i + 1)R) | v ∈ S_i}
    V_{i−1} ← V_i − ∪_{v∈S_i} ball_{V_i}(v, iR)
}
return C ;
```

FIG. 1. Cohen's algorithm for constructing a (k, R)-(roundtrip)-cover.

of roundtrip distance at most r from v in the subgraph $G' = G[V']$ induced by V'. In other words,

$$\text{ball}_{V'}(v, r) = \{u \in V' \mid \delta_{G'}(v \rightleftarrows u) \leq r\} .$$

We say that a set B is a ball centered at v if and only if $B = \text{ball}_{V'}(v, r)$, for some $V' \subseteq V$ and $r \geq 0$. Next, following Cohen [1999], we define:

Definition 2.4 ((k, R)-(*roundtrip*)-*cover*). A collection $\mathcal{C}$ of balls is a (k, R)-(roundtrip)-cover of a directed graph $G = (V, E)$ if and only if each ball in $\mathcal{C}$ is of radius at most kR, and for every $u, v \in V$ such that $\delta_G(u \rightleftarrows v) \leq R$, there is a ball $B \in \mathcal{C}$ such that $u, v \in B$.

For brevity, we usually refer to (k, R)-roundtrip-covers simply as (k, R)-covers. Figure 1 gives an extremely simple algorithm for constructing a (k, R)-roundtrip-cover of a graph. In the algorithm, **sample**(V_i, p) is a procedure that returns a random subset of V_i. Each element of V_i belongs to the returned set, independently, with probability p. The algorithm chooses collections of balls of decreasing radii. At the first iteration, when $i = k - 1$, about $n^{1/k}$ balls of radius kR are chosen. In the second iteration, with $i = k - 2$, about $n^{2/k}$ balls of radius $(k - 1)R$ are chosen, etc. In each iteration, when a ball $\text{ball}_{V_i}(v, (i + 1)R)$ is added to the cover $\mathcal{C}$, its *core* $\text{ball}_{V_i}(v, iR)$ is removed from the graph. The following theorems are adaptations, and slight improvements of results of Cohen [1999]. Our proofs are slightly different from the proofs given by Cohen, allowing us to get logarithmic improvements over her results. Our arguments are somewhat reminiscent of arguments used in Thorup and Zwick [2005].

THEOREM 2.5. *The collection $\mathcal{C}$ constructed by calling* **cover**(G, k, R) *is a (k, R)-roundtrip-cover of the* directed *graph $G = (V, E)$. For every $v \in V$, the* expected *number of balls in the cover containing v is at most $kn^{1/k}$.*

PROOF. We first argue that the set $\mathcal{C}$ generated by the algorithm is indeed a (k, R)-cover. Let $u, v \in V$ be such that $\delta_G(u \rightleftarrows v) \leq R$. Let c be a closed tour in G of total length at most R that contains u and v. Let i be the largest index for which a vertex u' on c is contained in the core $\text{ball}_{V_i}(w, iR)$ of a ball $\text{ball}_{V_i}(w, (i + 1)R)$ that is added to the cover. By the maximality of i, all the vertices on c are contained in V_i. Thus, $\delta_{G[V_i]}(w \rightleftarrows v) \leq \delta_{G[V_i]}(w \rightleftarrows u') + \delta_{G[V_i]}(u' \rightleftarrows v) \leq iR + R = (i + 1)R$, and $v \in \text{ball}_{V_i}(w, (i + 1)R)$. Similarly, $u \in \text{ball}_{V_i}(w, (i + 1)R)$, as required.

We now bound the expected number of balls in the cover that contain a given vertex $v \in V$. We show that, at each iteration, the expected number of balls containing v

that are added to the cover is at most $n^{1/k}$. Clearly, this is so in the first iteration, when $i = k - 1$, as then the expected number of balls added to the cover is only $n^{1/k}$.

Consider $B = \text{ball}_{V_{i+1}}(v, (i+1)R)$, the set of vertices that are at distance at most $(i+1)R$ from v at the beginning of the $(i+1)$-st iteration. (We number an iteration according to the value of the variable i at that iteration. Thus, the iteration following the $(i+1)$-st iteration would be the ith iteration.) Let $\ell = |B|$ be the number of these vertices. Each vertex $u \in B$ has a probability of $n^{-(i+1)/k}$ of being chosen to S_{i+1}. If a vertex u is chosen, then v would be contained in the core $\text{ball}_{V_{i+1}}(u, (i+1)R)$, and v would be removed from the graph and would not be contained in any ball added to the cover at the ith iteration. Note that, if $v \in \text{ball}_{V_i}(u, (i+1)R)$, then clearly $v \in \text{ball}_{V_{i+1}}(u, (i+1)R)$, and thus $u \in B$.

Thus, for every $u \in B$, the probability that u is chosen at the ith iteration and that $v \in \text{ball}_{V_i}(u, (i+1)R)$ is at most $(1 - n^{-(i+1)/k})^{\ell} \cdot n^{-i/k}$. (None of the vertices of B should be chosen at the $(i+1)$-st, i.e., previous, iteration, and u has to be chosen at the ith, i.e., current, iteration.) The expected number of balls that contain v in the ith iteration is therefore, at most $\ell(1 - n^{-(i+1)/k})^{\ell} \cdot n^{-i/k}$.

It is easy to check that the function $x(1-p)^x$, where $0 < p < 1$ is a fixed constant, attains it maximum value of $\frac{1}{\mathrm{e}p}$ when $x = -\frac{1}{\ln(1-p)}$. It follows easily that the expected number of balls containing v that are added to the cover at the ith iteration is at most $n^{1/k}/\mathrm{e}$, as required. □

THEOREM 2.6. *If the sampling probability in the ith iteration of algorithm* **cover** *is increased from $n^{-i/k}$ to $(n/\ln n)^{-i/k}$, then with very high probability, each vertex $v \in V$ is contained in at most $4kn^{1/k}(\ln n)^{1-1/k}$ balls of the collection $\mathcal{C}$ returned by the algorithm. The collection $\mathcal{C}$ is still a (k, R)-cover of G.*

PROOF. The proof is a relatively simple modification of the proof of Theorem 2.5. We show that at each iteration, with very high probability, for every vertex $v \in V$, the number of balls containing v that are added to the cover is at most $4n^{1/k}(\ln n)^{1-1/k}$. For brevity, we let $p_i = (\ln n/n)^{i/k}$ be the probability that a vertex is added to the sample S_i at the i-th iteration of algorithm **cover**.

The number of balls added to the cover in the first iteration, that is, when $i = k-1$, is a random variable distributed according to the binomial distribution $B(n, p_{k-1})$. Using the Chenoff bound, we easily get that the probability that this number is more than, say, $4n^{1/k}(\ln n)^{1-1/k}$ is exponentially small.

Let $v \in V$. To bound the number of balls containing v that are added to the cover in the ith iteration, we consider again the ball $B = \text{ball}_{V_{i+1}}(v, (i+1)R)$. Let $\ell_{i+1} = 2n^{(i+1)/k}(\ln n)^{1-(i+1)/k}$. Note that $p_{i+1}\ell_{i+1} = 2\ln n$. We consider two cases. If $|B| \geq \ell_{i+1}$, then with a probability of at least $1 - n^{-2}$, at least one of the vertices of B is added to S_{i+1}. This is so because

$$(1 - p_{i+1})^{\ell_{i+1}} \leq \exp(-p_{i+1}\ell_{i+1}) \leq n^{-2}.$$

If a vertex from $u \in B$ is sampled, then the core $\text{ball}_{V_{i+1}}(u, (i+1)R)$, which contains v, is removed from the graph and v would not be contained in any additional ball.

Assume therefore that $|B| < \ell_{i+1}$ and that $B \cap S_{i+1} = \phi$. The number of balls added to the cover in the i-iteration that contain v is at most $|B \cap S_i|$, the number of vertices in B that are sampled in the i-iteration. This is a random variable that is

stochastically dominated by a random variable distributed according to the binomial distribution $B(\ell_{i+1}, p_i)$. The expectation of this random variable is $\mu = p_i \ell_{i+1} = 2n^{1/k}(\ln n)^{1-1/k}$. Using the Chernoff bound, we easily get that the probability that this number exceeds, say, $2\mu = 4n^{1/k}(\ln n)^{1-1/k}$ is again exponentially small. This completes the proof of the theorem. □

Definition 2.7 (IN AND OUT TREES). Let $G = (V, E)$ be a weighted directed graph. Let $V' \subseteq V$, $v \in V$ and $r \geq 0$. Let $B = \text{ball}_{V'}(v, r)$. We let OutTree$(B, v)$ be a tree containing directed shortest paths in $G' = G[V']$ from v to all the vertices of B. Similarly, we let InTree(B, v) be a tree containing directed shortest paths from all the vertices of B to v. We let InOutTrees$(B, v) =$ InTree$(B, v) \cup$ OutTree(B, v). We refer to InOutTrees(B, v) as a *double-tree*. We sometimes omit the center v from these notations when it is clear from the context.

Let $B = \text{ball}_{V'}(v, r)$. It is easy to check that if $u \in B$, and w is on a shortest path from u to v, or from v to u in $G' = G[V']$, then $w \in B$. This follows from the fact that $\delta_{G'}(v \rightleftarrows w) \leq \delta_{G'}(v \rightleftarrows u)$. Thus, all the vertices in InTree(B) and OutTree(B), and thus in InOutTrees(B), are in the ball B. It follows that the number of edges in InOutTrees(B) is at most $2(|B| - 1)$. (Note that InTree(B) and OutTree(B) may share edges.) We also need the following lemma:

LEMMA 2.8. *Let $B = ball_{V'}(v, r)$, where $V' \subseteq V$,and let $u_1, u_2 \in B$. Then,* InOutTrees(B, v) *contains a closed directed tour containing u_1 and u_2 of length at most $2r$.*

PROOF. Let $G' = G[V']$. By definition, InOutTrees(B, v) contained a closed tour of length $\delta_{G'}(v \rightleftarrows u_1)$ containing v and u_1, and a closed tour of length $\delta_{G'}(v \rightleftarrows u_2)$ containing v and u_2. By combining these tours we get a tour of length at most $\delta_{G'}(v \rightleftarrows u_1) + \delta_{G'}(v \rightleftarrows u_2) \leq 2r$ containing u_1 and u_2. □

We can now present a proof of Theorem 2.3:

PROOF (OF THEOREM 2.3). Let $\epsilon' = \frac{\epsilon}{2k}$. For every $1 \leq i \leq \log_{1+\epsilon'}(2nW)$, let $\mathcal{C}_i$ be a (k, R_i)-cover of G, where $R_i = (1+\epsilon')^i$, constructed by calling **cover**(G, k, R_i). Let H be composed of the union of InOutTrees(B), for every $B \in \cup_i \mathcal{C}_i$. Clearly, the obtained graph is a $(2k + \epsilon)$-roundtrip-spanner of G and the *expected* number of edges in it is $O((k^2/\epsilon)n^{1+1/k} \log(nW))$. □

Our roundtrip-spanner construction actually proves a stronger result:

Definition 2.9 (DOUBLE-TREE COVERS). Let $G = (V, E)$ be a weighted directed graph. A collection $\mathcal{T}$ of double-trees of G is said to be a t-double-tree-cover of G if and only if, for every $u, v \in V$, there is a double-tree $T \in \mathcal{T}$ such that $u, v \in T$ and $\delta_T(u \rightleftarrows v) \leq t \cdot \delta_G(u \rightleftarrows v)$.

As the roundtrip spanners constructed in the proof of Theorem 2.3 are composed of double-trees corresponding to collections of balls that cover the graph, we immediately get:

THEOREM 2.10. *For every integer $k \geq 1$ and every $\epsilon > 0$, every weighted directed graph on n vertices with edge weights taken from $[1, W]$ has a $(2k + \epsilon)$-double-tree-cover such that every vertex $v \in V$ is contained in at most $O((k^2/\epsilon)\, n^{1/k}\, (\log n)^{1-1/k} \log(nW))$ double-trees.*

```
algorithm spanner(G, k, ε)

F ← φ ; ε' ← ε/(6k) ; A ← n^3
for i ← 0 to ⌈3 log_{1+ε'} n⌉
{
   G_0 ← G
   for j ← 0 to ⌈log_{n^3}(2nW)⌉
   {
      C_j ← cover(G_j, k, (1 + ε')^i A^j)
      F ← F ∪ (∪_{B∈C_j} InOutTrees(B))
      G_{j+1} ← contract(G_j, C_j)
   }
}
return (V, F) ;
```

FIG. 2. Constructing a $(2k + \epsilon)$-roundtrip-spanner of a directed graph.

3. *Removing the Dependence on the Edge Weights*

In the previous section, we showed that any directed graph on n vertices with edge weights in the range $[1, W]$ has a $(2k + \epsilon)$-roundtrip-spanner with at most $O((k^2/\epsilon)\, n^{1+1/k}\, \log (nW))$ edges. In this section, we show that the dependence of the number of edges in the spanner on W, the largest edge weight can be removed. Algorithm **spanner**, described in Figure 2, constructs a $(2k+\epsilon)$-roundtrip-spanner of a weighted directed graph that contains only $O((k/\epsilon)^2\, n^{1+1/k}\, (\log n)^{2-1/k})$ edges. We still assume, however, that the edge weights in the graph are in the range $[1, W]$.

We should note, however, that the dependence on W becomes an issue only when W is extremely large, that is, super-polynomial in n. Some edges may then belong to $\Omega(\log \Delta / \log (1 + \epsilon))$ of the covers constructed in the proof of Theorem 2.3, where $\Delta = 2nW$ is a bound on the largest finite roundtrip distance in the graph.

It is not too difficult, however, to remove this dependence on W. To illustrate the basic idea used to accomplish this, suppose for a moment that the graph is undirected, and that we are only considering distances in the range $[R, 2R)$. Then clearly, we do not need to consider edges of weight larger than $2R$. Also, with a marginal decrease of distances, we can *contract* all edges of length $o(R/n)$. As a consequence, an edge of length w is only considered for distances between w and $O(nw)$. Thus, when exponentially increasing distances are considered, the number of times each edge is considered is logarithmic in n. Below, we get this basic idea to work for our roundtrip spanners in directed graphs.

Algorithm **spanner** is fairly similar to the algorithm sketched in the proof of Theorem 2.3 for the construction of roundtrip-spanners. The main difference is that balls that are contained in a (k, R_{ij})-cover, were $R_{ij} = (1 + \epsilon')^i A^j$ and $A = n^3$, are now *contracted* before the next cover is constructed. The contraction is peformed by a call to algorithm **contract**. Algorithm **contract** receives a graph G and a collection of balls $\mathcal{C}$. The vertices of each ball $B \in \mathcal{C}$ are merged into a single vertex. Nondisjoint balls are also merged together. Thus, if, for example, $B_1, B_2, \ldots, B_\ell \in \mathcal{C}$ and for every $1 \leq i < j \leq \ell$ there are indices $i = i_1 < i_2 < \cdots < i_r = j$ such that $B_{i_j} \cap B_{i_{j+1}} \neq \phi$, for $1 \leq j < r$, then all the vertices of $\cup_{i=1}^{\ell} B_i$ are merged into a single vertex. After these merge operations, internal

edges are removed. It is also possible to replace each set of parallel edges by the cheapest edge in this set, but this is not essential. Each nonremoved edge retains its original endpoints in the original graph. Thus, edges added to the set F of edges are considered to be edges of the original graph.

To simplify the analysis of algorithm **spanner**, we assume that it uses the version of algorithm **cover** in which the sampling probability at the ith iteration is slightly raised to $(n/\ln n)^{-i/k}$. As shown in Theorem 2.6, with high probability, each vertex of the graph is contained in at most $4kn^{1/k}(\ln n)^{1-1/k}$ balls of the produced cover. We further assume that each vertex is indeed contained in at most this number of balls, as otherwise, we may simply run algorithm **cover** again.

THEOREM 3.1. *Let $k \geq 1$ be an integer, let $\epsilon > 0$, and let G be a weighted directed graph on n vertices with edge weights in the range $[1, W]$. Then, a call to algorithm* **spanner** *produces a $(2k + \epsilon)$-roundtrip-spanner of G with $O((k/\epsilon)^2\, n^{1+1/k}\, (\log n)^{2-1/k})$ edges.*

PROOF. We begin by bounding the number of edges in the subgraph constructed by the algorithm. Let $I = \lceil 3 \log_{1+\epsilon'} n \rceil = O((k/\epsilon) \log n)$ be the number of iterations of the outer loop of **spanner**, and let $J = \lceil \log_{n^3}(2nW) \rceil$ be the number of iteration of the inner loop of **spanner**. We show that in each iteration of the outer loop, the number of edges added to F is $O((k/\epsilon)\, n^{1+1/k}\, (\log n)^{1-1/k})$. The total number of edges is therefore $O((k/\epsilon)^2\, n^{1+1/k}\, (\log n)^{2-1/k})$, as required.

Let $D = 4kn^{1/k}(\ln n)^{1-1/k}$ be the bound, obtained in Theorem 2.6, on the number of balls in a cover that may contain a given vertex. At the start of each iteration of the outer loop the graph G_0 is initialized to be G. For the sake of the analysis, we assign each vertex of G_0 at that stage D units of credit. Each unit of credit can pay for the inclusion of two edges in the spanner. When the vertices of G_j are merged by algorithm **contract**, producing G_{j+1}, the unused credits of the vertices of G_j are passed on to the vertices of G_{j+1}. We show that at each stage, the vertices of G_j can pay for the inclusion of all the edges added to the spanner at the jth iteration of the inner loop, without ever running out of credits. It would follow then that the number of edges added to the spanner in each iteration of the outer loop is at most $O(nD) = O((k/\epsilon)\, n^{1+1/k}\, (\log n)^{1-1/k})$.

To show that the vertices of the graphs G_j, for $1 \leq j \leq J$, never run out of credit, we show that, when the graph G_j is formed, each of its vertices has at least D units of credit. This clearly holds for $j = 0$. We then show that, if each vertex of G_j starts with D credits, then so will each vertex of G_{j+1}.

Algorithm **cover** constructs in each graph G_j a cover $\mathcal{C}_j$. For each ball $B \in \mathcal{C}_j$, a double-tree InOutTrees(B) is added to the spanner. This double tree contains $2(|B| - 1)$ edges. As each vertex is contained in at most D balls of the cover, the vertices of G_j clearly have enough credit to pay for the addition of these double-trees to the spanner. Furthermore, we show that there is enough "leftover" credit to assign each vertex of G_{j+1} with at least D units of credit.

Let $B_1, B_2, \ldots, B_\ell \in \mathcal{C}_j$ be a collection of balls that are all merged into a single vertex of G_{j+1}. The vertices of $\cup_{i=1}^{\ell} B_i$ have, altogether, at least $D\,|\cup_{i=1}^{\ell} B_i|$ units of credit. Of these, $\sum_{i=1}^{\ell}(|B_i| - 1)$ units are used to pay for the edges of the double-trees. To show that the vertex of G_{j+1} fomred by merging $B_1, B_2, \ldots, B_\ell$ inherits at least D credits, we have to show that $D\,|\cup_{i=1}^{\ell} B_i| - \sum_{i=1}^{\ell}(|B_i| - 1) \geq D$, or

equivalently, that

$$D\left(|\cup_{i=1}^{\ell} B_i| - 1\right) \;\geq\; \sum_{i=1}^{\ell}(|B_i| - 1)\ .$$

To show that this inequality holds, we argue as follows: Let d be the maximum number of balls, out of $B_1, B_2, \ldots, B_\ell$, that all contain the same vertex. Clearly, $d\,|\cup_{i=1}^{\ell} B_i| \geq \sum_{i=1}^{\ell} |B_i|$. As $d \leq \ell$, and $d \leq D$, we get that

$$D\left(|\cup_{i=1}^{\ell} B_i| - 1\right) \;\geq\; d\left(|\cup_{i=1}^{\ell} B_i| - 1\right) \;\geq\; \sum_{i=1}^{\ell}(|B_i| - 1)\ ,$$

as required.

We next have to bound the stretch of the spanner constructed by **spanner**. We focus now on the ith iteration of the outer loop of algorithm **spanner**. For brevity, we supress the index i from the notations introduced below. Let V_j be the vertex set of G_j. Each vertex of G_j corresponds to a set of vertices in the original graph $G = G_0$. If $v \in V$ is a vertex in G, we let v_j denote the vertex of G_j that "absorbed" v. We let $F_j = \cup_{B \in \mathcal{C}_j} \text{InOutTrees}(B)$ denote the edges added to F, that is, to the spanner, at the jth iteration of the inner loop of **spanner**.

Define a sequence $d_0 = 0$ and $d_{j+1} = 2n(kR_{ij} + d_j)$, for $0 \leq j \leq J$. We claim:

CLAIM 3.2. *If $u, v \in V$ and $u_j = v_j$, that is, u and v are absorbed into the same vertex of G_j, then $\delta_F(u \rightleftarrows v) \leq d_j$.*

PROOF. The proof is by induction on j. For $j = 0$, the claim is obvious. Let $j \geq 0$, and suppose that the claim holds for j. We show that it also holds for $j+1$. Let $u, v \in V$ such that $u_{j+1} = v_{j+1}$. Then, $\mathcal{C}_j$ contains balls $B_1, \ldots, B_\ell$, where $\ell \leq n$, such that $u_j \in B_1$, $v_j \in B_\ell$, and $B_r \cap B_{r-1} \neq \phi$, for $2 \leq r \leq \ell$. Let $w_1 = u_j$, $w_{\ell+1} = v_j$, and for $2 \leq r \leq \ell$, let w_r be an arbitrary vertex from $B_r \cap B_{r-1}$. As $w_r, w_{r+1} \in B_r$, we get that $\delta_{F_j}(w_r \rightleftarrows w_{r+1}) \leq 2kR_{ij}$. It follows that

$$\delta_{F_j}(u_j \rightleftarrows v_j) \;=\; \delta_{F_j}(w_1 \rightleftarrows w_{\ell+1}) \;\leq\; \sum_{r=1}^{\ell} \delta_{F_j}(w_r \rightleftarrows w_{r+1}) \;\leq\; 2knR_{ij}.$$

We now want to translate this upper bound on $\delta_{F_j}(u_j \rightleftarrows v_j)$ into an upper bound on $\delta_F(u, v)$. We claim that $\delta_F(u \rightleftarrows v) \leq \delta_{F_j}(u_j \rightleftarrows v_j) + 2nd_j$. To show this, we argue as follows: Let P be the shortest roundtrip path from u_j to v_j in (V_j, F_j). Let $e_1, e_2, \ldots, e_t$ be the edges on this path. Note that $t \leq 2(n-1)$. Each one of these edges corresponds to an edge in the original graph. Let $e_r = (x_r, y_r)$, for $1 \leq r \leq t$, be the endpoints of these edges in G. We then have $(y_r)_j = (x_{r+1})_j$, for $1 \leq r \leq t$, where we let $x_{t+1} = x_1$. We also have $(x_1)_j = u_j$, and $(x_p)_j = (y_{p-1})_j = v_j$, for some $1 \leq p \leq t$. It follows that

$$\delta_F(x_1 \rightleftarrows x_p) \;\leq\; \sum_{r=1}^{t} \delta_F(x_r \to y_r) + \sum_{r=1}^{t} \delta_F(y_r \to x_{r+1}).$$

Now, $\sum_{r=1}^{t} \delta_F(x_r \to y_r)$ is just the length of P, that is, $\delta_{F_j}(u_j \rightleftarrows v_j)$. As $(y_r)_j = (x_{r+1})_j$, for $1 \leq r \leq t$, we get by the inductive hypothesis that $\delta_F(y_r \to x_{r+1}) \leq$

$\delta_F(y_r \rightleftarrows x_{r+1}) \leq d_j$. It follows, therefore, that

$$\delta_F(x_1 \rightleftarrows x_p) \leq \delta_{F_j}(u_j \rightleftarrows v_j) + 2(n-1)d_j.$$

As $u_j = (x_1)_j$ and $v_j = (x_p)_j$, we finally get that

$$\delta_F(u \rightleftarrows v) \leq \delta_F(u \rightleftarrows x_1) + \delta_F(x_1 \rightleftarrows x_p) + \delta(x_p \rightleftarrows v) \leq \delta_{F_j}(u_j \rightleftarrows v_j) + 2nd_j.$$

Combining this with $\delta_{F_j}(u_j \rightleftarrows v_j) \leq 2knR_{ij}$, we get

$$\delta_F(u \rightleftarrows v) \leq 2knR_{ij} + 2nd_j = 2n(kR_{ij} + d_j) = d_{j+1},$$

as required. □

CLAIM 3.3. *The sequence* $d_0 = 0$ *and* $d_{j+1} = 2n(kR_{ij} + d_j)$, *for* $j \geq 0$, *satisfies* $d_j \leq 4k(n/A)R_{ij}$, *for* $j \geq 1$.

PROOF. Recall that $R_{ij} = (1+\epsilon')^i A^j$ and that $A = n^3$. It is easy to prove, by induction, that $d_j = kR_{ij} \sum_{r=1}^{j} (\frac{2n}{A})^r$, for $j \geq 1$. As $\sum_{r=1}^{j} (\frac{2n}{A})^r \leq \frac{4n}{A}$, for $n \geq 2$, we get that $d_j \leq 4k(n/A)R_{ij}$, for $j \geq 1$. □

We can now finally bound the stretch of the spanner constructed by algorithm **spanner**. Let $u, v \in V$ be two vertices of G. Clearly, there exist $0 \leq i \leq I$ and $0 \leq j \leq J$ such that $(1+\epsilon')^{-1} R_{ij} \leq \delta_G(u \rightleftarrows v) \leq R_{ij}$, where $R_{ij} = (1+\epsilon')^i A^j$. Consider the (i, j)-th iteration of the algorithm. As $\delta_G(u \rightleftarrows v) \leq R_{ij}$, we clearly have $\delta_{G_j}(u_j \rightleftarrows v_j) \leq R_{ij}$. Thus, the cover $\mathcal{C}_j$ must contain a ball $B \in \mathcal{C}_j$ such that $u_j, v_j \in B$ and thus $\delta_{F_j}(u_j \rightleftarrows v_j) \leq 2kR_{ij}$. If $j = 0$, then we get that $\delta_F(u, v) \leq 2kR_{ij}$. Assume, therefore, that $j \geq 1$. Using the same arguments, we have used in the proof of Claim 3.2, and using the bound on d_j, we get

$$\delta_F(u \rightleftarrows v) \leq \delta_{F_j}(u_j \rightleftarrows v_j) + 2nd_j \leq 2kR_{ij} + 8k(n^2/A)R_{ij} \leq 2k(1+\epsilon')R_{ij}.$$

Thus, the stretch of the spanner is at most $2k(1+\epsilon')^2 \leq 2k + \epsilon$, as required.

4. *Roundtrip Routing Schemes*

Fraigniaud and Gavoille [2001] and Thorup and Zwick [2001] describe an extremely efficient way of routing on trees:

THEOREM 4.1 (FRAIGNIAUD AND GAVOILLE 2001; THORUP AND ZWICK 2001). *Let* $T = (V, E)$ *be an* undirected *tree on* n *vertices with each edge* $e \in E$ *assigned a unique* $O(\log n)$*-bit* port number. *Then, it is possible to efficiently assign each vertex* $v \in V$ *an* $O(\log^2 n / \log\log n)$*-bit label, denoted label*(v), *such that if* $u, v \in V$, *then given label*(u) *and label*(v), *and nothing else, it is possible to find, in constant time, the port number assigned to the first edge on the path in* T *from* u *to* v.

Let B be a ball centered at v. We can convert each double-tree InOutTrees(B, v) into a standard tree InOutTree(B, v) (note the slight difference in notation) by considering InTree(B, v) and OutTree(B, v) as being trees on different sets of vertices, and joining them at v. For every $u \in B$, $u \neq v$, we let u' be the copy of u in the InTree(B, v), and u'' be the copy of u in OutTree(B, v). We also let $v' = v'' = v$. For every $u_1, u_2 \in B$, there is a directed path in InOutTree(B, v) from u_1' to u_2''. Although Theorem 4.1 was stated for undirected trees, it is easy

to extend it to directed trees like InOutTree(B, v), and thus to double-trees like InOutTrees(B, v).

Theorem 2.10 essentially reduces the problem of roundtrip routing in general directed graphs into the problem of routing in double-trees. Routing in double-trees could be done using the techniques of Fraigniaud and Gavoille [2001] and Thorup and Zwick [2001].

One big problem still remains, however. Suppose that u wants to route a message to v. We know that the double-tree cover $\mathcal{T}$ contains a double-tree T in which $\delta_T(u \rightleftarrows v) \leq (2k+\epsilon)\cdot\delta_G(u \rightleftarrows v)$. But, how does u *identify* this tree?

For any $R_j = (1+\epsilon)^j$, where $1 \leq j \leq \log_{1+\epsilon}(nW)$, and any $0 \leq i \leq k-1$, the double-tree cover $\mathcal{T}$ of Theorem 2.10 contains at most $\tilde{O}(n^{1/k})$ double-trees of radius $(i+1)R_j$ containing v. A naive solution to the problem posed above would be to specify all these trees in $label(v)$. But $label(v)$ would then be of length $\Omega(kn^{1/k}\log_{1+\epsilon}(nW))$, which is too much, as we are aiming for labels of polylogarithmic size.

We can decrease the labels to polylogarithmic size at the price of doubling the resulting stretch. We need the notion of *extended core* and the following lemma:

LEMMA 4.2. *Let $\mathcal{C}$ be a (k, R)-(roundtrip)-cover of a weighted directed graph $G = (V, E)$ constructed using algorithm* **cover** *of Figure* 1. *Let $v \in V$. Let i be the largest index for which there is a ball $B = ball_{V_i}(w, (i+1)R) \in \mathcal{C}$, such that $v \in B' = ball_{V_i}(w, (i+1/2)R) \in \mathcal{C}$. (We refer to B' as the* extended core *of B.) Then, if $u \in V$ and $\delta_G(u \rightleftarrows v) \leq R/2$, then $u \in B$.*

PROOF. Let c be a shortest closed tour containing u and v. We show that $c \subseteq V_i$, and therefore, $\delta_{G[V_i]}(w \rightleftarrows u) \leq \delta_{G[V_i]}(w \rightleftarrows v) + \delta_{G[V_i]}(v \rightleftarrows u) \leq (i+1/2)R + R/2 \leq (i+1)R$, and thus $u \in B$, as required.

Suppose, therefore, for the sake of contradiction, that there is a vertex u' on c that is not in V_i. This means that there is a vertex $w' \in S_j$, for $j > i$ such that $u' \in \text{ball}_{V_j}(w', jR)$. But then $\delta_{G[V_j]}(w' \rightleftarrows v) \leq \delta_{G[V_j]}(w' \rightleftarrows u') + \delta_{G[V_j]}(u' \rightleftarrows v) \leq jR + R/2 \leq (j+1/2)R$, a contradiction to the maximality of i. □

Let $\epsilon' = \frac{\epsilon}{4k}$. Let $v \in V$ be a vertex. For every $R_j = (1+\epsilon')^j$, where $1 \leq j \leq \ell = \lceil \log_{1+\epsilon}(nW) \rceil$, we choose a *single* ball $\text{ball}_{V_i}(w, (i+1)R_j) \in \mathcal{C}$, such that $v \in \text{ball}_{V_i}(w, (i+1/2)R_j) \in \mathcal{C}$, as in Lemma 4.2, and let $cent_j(v) = w$ be its center. We let

$$label_j(v) = (cent_j(v), tree\text{-}label_j(v)),$$

where $tree\text{-}label_j(v)$ is the label of v in the double-tree of the ball centered at $cent_j(v)$, assigned by the tree routing scheme of Theorem 4.1. Finally, we let

$$label(v) = (label_1(v), label_2(v), \ldots, label_\ell(v)).$$

The routing table $RT(u)$ of a vertex $u \in V$ is composed of $\ell = \lceil \log_{1+\epsilon}(nW) \rceil$ (two-level) hash tables. The jth table $RT_j(u)$ holds the centers of the balls of the (k, R_j)-cover that contain u. (It is possible to combine these tables into a single hash table, but for simplicity, we assume that they are separate.) For each one of these centers, $RT_j(u)$ holds the tree-label of u in the corresponding double-tree.

Suppose now that u wants to send a message to v. It simply finds the smallest j for which $cent_j(v) \in RT_j(u)$. The message can then be routed on the double-tree centered at $cent_j(v)$. The label of u in that tree is extracted from $RT_j(u)$, and the label of v in that tree is extracted from $label_j(v)$.

The initial routing decision takes $O(\log_{1+\epsilon}(nW))$ time. Each subsequent decision takes only constant time, as the correct double-tree was already identified. An implementation of linear space, constant query time, hash tables is described in Fredman et al. [1984]. (For deterministic constructions of such hash tables, see Alon and Naor [1996] and Hagerup et al. [2001].) We thus have:

THEOREM 4.3. *The roundtrip routing scheme described above has stretch* $4k+\epsilon$. *It uses local routing tables of size* $\tilde{O}(\frac{k}{\epsilon}n^{1/k}\log(nW))$ *and* $o(\frac{k}{\epsilon}\log^2 n\log(nW))$*-bit labels. The headers attached to the messages are* $o(\log^2 n)$*-bit long. The initial routing decision takes* $O(\frac{k}{\epsilon}\log(nW))$ *time. Each subsequent routing decision takes only constant time.*

PROOF. Let $u, v \in V$ and suppose that $\delta_G(u \rightleftarrows v) = R/2$, where $(1+\epsilon')^{j-1} \leq R \leq (1+\epsilon')^j$. Consider the ball B, centered at $cent_j(v)$ that contains v in its extended core. By Lemma 4.2, this ball also contains u. The radius of this ball is at most $k(1+\epsilon')^j$. The double-tree corresponding to this ball contains a closed tour containing u and v of total length at most $2k(1+\epsilon')^j$. Messages from u to v, and back, would be routed along this tour, giving a stretch of at most $(1+\epsilon')4k = 4k+\epsilon$. □

5. *A Roundtrip Routing Scheme with Stretch* 3

In this section, we describe an essentially optimal roundtrip routing scheme of stretch 3. It uses local routing tables of size $\tilde{O}(n^{1/2})$ and $O(\log n)$-bit labels. Each routing decision takes constant time. This routing scheme combines ideas from Cowen [2001], Cowen and Wagner [1999, 2000], and Thorup and Zwick [2001]. We start with the following definitions:

Definition 5.1 (*Roundtrip Ordering*). Let $G = (V, E)$ be a weighted directed graph and let $v \in V$. We assume, without loss of generality, that $V = \{1, 2, \ldots, n\}$. We say that $u_1 \prec_v u_2$ if and only if one of the following conditions holds:

(1) $\delta(u_1 \rightleftarrows v) < \delta(u_2 \rightleftarrows v)$,
(2) $\delta(u_1 \rightleftarrows v) = \delta(u_2 \rightleftarrows v)$ but $\delta(u_1 \rightarrow v) < \delta(u_2 \rightarrow v)$,
(3) $\delta(u_1 \rightleftarrows v) = \delta(u_2 \rightleftarrows v)$ and $\delta(u_1 \rightarrow v) = \delta(u_2 \rightarrow v)$ but $u_1 < u_2$.

We say that $u_1 \preceq u_2$ if and only if $u_1 \prec u_2$ or $u_1 = u_2$. It is easy to verify that $\prec_v$ and $\preceq_v$ are transitive relations.

Definition 5.2 (*Centers and Clusters*). Let $G = (V, E)$ be a weighted directed graph. Let $A \subseteq V$ be a set of *centers*. For every $v \in V$, we let $cent_A(v)$ be the element of A that satisfies $cent_A(v) \preceq_v w$, for every $w \in A$. The cluster $C_A(u)$ of a vertex $u \in V$ is defined as follows:

$$C_A(u) = \{v \in V \mid u \prec_v cent_A(v)\}$$

Finally, we let $\delta(A \rightleftarrows v) = \delta(cent_A(v) \rightleftarrows v)$.

With these careful definitions, we now have:

LEMMA 5.3. *If $v \in C_A(u)$ and w is on a shortest path from u to v, then $v \in C_A(w)$.*

PROOF. If $w = u$, then the claim is obvious. Otherwise, as w is on a shortest path from u to v, we have $\delta(w \rightleftarrows v) \leq \delta(u \rightleftarrows v)$ and $\delta(w \to v) < \delta(u \to v)$. (Note that we assume that all edge weights are positive.) Thus, $w \prec_v u$. As $v \in C_A(u)$, we get that $u \prec_v cent_A(v)$. By the transitivity of $\prec_v$, it follows that $w \prec_v cent_A(v)$, and thus $v \in C_A(w)$, as required. □

Using a simple adaptation of an iterative sampling technique of Thorup and Zwick [2001] to the directed case, we obtain:

THEOREM 5.4. *Let $G = (V, E)$ be a weighted directed graph with $|v| = n$ and $|E| = m$. It is possible, in $O(mn)$ time, to find a set $A \subseteq V$ of centers such that $|A| = O((n \log n)^{1/2})$ and $|C_A(w)| = O((n \log n)^{1/2})$, for every $w \in V$.*

The stretch 3 roundtrip routing schemes starts by choosing a set of centers A that satisfies the conditions of Theorem 5.4. Each vertex $v \in V$ is then assigned the following label:

$$label(v) = (v, cent_A(v), port(cent_A(v), v)),$$

where for every $u, v \in V$, we let $port(u, v)$ be the port number corresponding to the first edge on the shortest path from u to v in G.

The routing table $RT(u)$ at a vertex $u \in V$ is a (2-level) hash table that holds for each $v \in A \cup C_A(u)$ the pair $(v, port(u, v))$. The size of this table, which is linear in $|A \cup C_A(u)|$, is $O((n \log n)^{1/2})$ and for each $v \in A \cup C_A(u)$, the pair $(v, \text{port}(u, v))$ can be located in *worst-case* constant time. As before, we use the data structure of Fredman et al. [1984]. (See also Alon and Naor [1996] and Pagh [2000].)

When a message destined for v reaches u, it is routed as follows:

(0) If $u = v$, the message reached its destination.
(1) Otherwise, if $v \in A \cup C_A(u)$, route the message using the edge with port number $port(u, v)$. (This port number is found, in constant time, by accessing $RT(u)$.)
(2) Otherwise, if $u = cent_A(v)$, route along the edge with port number $port(cent_A(v), v)$. (The center $cent_A(v)$ and the port number $port(cent_A(v), v)$ are taken from $label(v)$.)
(3) Otherwise, route along the edge with port number $port(u, cent_A(v))$. (As $cent_A(v) \in A$, this port number can be obtained, in constant time, by accessing $RT(u)$.)

We now claim:

THEOREM 5.5. *The routing algorithm described above routes a package from u to v along a path whose length is most $\delta(u \to v) + \delta(u \rightleftarrows v)$. The stretch of the roundtrip route from u to v and back is, therefore, at most 3.*

The proof is very similar to proofs given in Cowen [2001] and Thorup and Zwick [2001] and it is brought here for the sake of completeness.

PROOF. We start by showing that the routing algorithm routes a package from u to v along a path whose length is most $\delta(u \rightleftarrows v) + \delta(u \to v)$. We consider three cases that correspond to the steps (1)–(3) of the routing algorithm.

Case 1. $v \in A \cup C_A(u)$. In step (2), the routing algorithm routes the package on the first edge $u \to u'$ of a shortest path from u to v. Also, we have $v \in A \cup C_A(u')$. If $v \in A$, the claim is obvious. If $v \in C_A(u)$, the claim follows from Lemma 5.3. By induction, we get therefore, that the message is routed, in this case, along a shortest path from u to v whose length is $\delta(u \to v)$.

Case 2. $u = cent_A(v)$. The algorithm routes the package out from u along the edge with the port number port($\text{cent}_A(v)$, v), i.e., on the first edge $u \to u'$ of a shortest path from $u = \text{cent}_A(v)$ to v. By Definition 5.1 we then have $u' \prec_v \text{cent}_A(v)$ and hence, by the Definition 5.1, we have $v \in C_A(u')$. By Case 1, the packet is then routed from u' to v along a shortest path. Thus, the packet is again routed from u to v along a shortest path whose length is $\delta(u \to v)$.

Case 3. $v \notin A \cup C_A(u)$ and $u \neq cent_A(v)$. The algorithm routes the package on the first edge of a shortest path from u to $cent_A(v)$. We consider two subcases.

Subcase 1. The packet eventually arrives to $cent_A(v)$. The path followed by the packet from u to $cent_A(v)$ is then a shortest path, as every vertex in the graph holds routing information to $cent_A(v)$. When the packet arrives to $cent_A(v)$ we are in Case 2 and the packet will be routed from $cent_A(v)$ to v along a shortest path. Thus, the total length of the route followed by the packet from u to v in this case is $\delta(u \to cent_A(v)) + \delta(cent_A(v) \to v)$.

Subcase 2. The packet arrives to a vertex w on the shortest path between u to $cent_A(v)$ for which $v \in C_A(w)$. By Case 1, the packet is then routed from w to v along a shortest path. The total length of the path is again bounded by $\delta(u \to cent_A(v)) + \delta(cent_A(v) \to v)$.

Now,

$$\begin{aligned}\delta(u \to cent_A(v)) + \delta(cent_A(v) \to v) &\leq \delta(u \to v) + \delta(v \to cent_A(v)) + \delta(cent_A(v) \to v) \\ &= \delta(u \to v) + \delta(cent_A(v) \rightleftarrows v) \\ &\leq \delta(u \to v) + \delta(u \rightleftarrows v),\end{aligned}$$

where the first inequality follows from the triangle inequality, and the last inequality follows as $v \notin C_A(u)$ and thus $cent_A(v) \preceq_v u$.

Similarly, the routing algorithm routes a packet from v to u along a path of length at most $\delta(v \to u) + \delta(v \rightleftarrows u)$. The total length of the roundtrip path followed by the packet from u to v and back is therefore at most

$$\delta(u \to v) + \delta(u \rightleftarrows v) + \delta(v \to u) + \delta(v \rightleftarrows u) \leq 3\delta(u \rightleftarrows v),$$

as required. □

6. *Concluding Remarks*

We have made explicit the notion of *roundtrip-spanners* and obtained substantially improved algorithms for constructing them. In particular, we showed that for any fixed integer $k \geq 1$ and any fixed $\epsilon > 0$, any directed graph on n vertices with edge weights in the range $[1, W]$ has a $(2k + \epsilon)$-roundtrip-spanner with $O((k^2/\epsilon)n^{1+1/k} \log(nW))$ edges, and that such a spanner can be constructed in $\tilde{O}(mn)$ time. We also showed that each such graph has a $(2k+\epsilon)$-roundtrip-spanner with $O((k/\epsilon)^2 n^{1+1/k} (\log n)^{2-1/k})$ edges. An interesting open problem is whether

it is possible to reduce the stretch in this construction to $2k - 1$, which is believed to be optimal even for undirected graphs.

We have also obtained substantially improved roundtrip routing schemes. To obtain logarithmic size labels, we had to double the stretch of the schemes from $2k + \epsilon$ to $4k + \epsilon$. Another interesting open problem is whether there is a way of avoiding this doubling, possibly using handshaking.

Finally, Cohen and Zwick [2001] describe an $\tilde{O}(n^2)$ time algorithm for computing stretch 3 distances between all pairs of vertices of an n-vertex *undirected* graph. Is it possible to obtain an analog result for roundtrip distances in *directed* graphs?

REFERENCES

ALON, N., AND NAOR, M. 1996. Derandomization, witnesses for Boolean matrix multiplication and construction of perfect hash functions. *Algorithmica 16*, 434–449.

ALTHÖFER, I., DAS, G., DOBKIN, D., JOSEPH, D., AND SOARES, J. 1993. On sparse spanners of weighted graphs. *Disc. Comput. Geom. 9*, 81–100.

AWERBUCH, B., BERGER, B., COWEN, L., AND PELEG, D. 1999. Near-linear time construction of sparse neighborhood covers. *SIAM J. Comput. 28*, 263–277. (A preliminary version appears at the *Proceedings of FOCS'93.*)

COHEN, E. 1999. Fast algorithms for constructing t-spanners and paths with stretch t. *SIAM J. Comput. 28*, 210–236.

COHEN, E., AND ZWICK, U. 2001. All-pairs small-stretch paths. *J. Algor. 38*, 335–353.

COWEN, L. 2001. Compact routing with minimum stretch. *J. Algor. 38*, 170–183.

COWEN, L., AND WAGNER, C. 1999. Compact roundtrip routing in digraphs. In *Proceedings of the 10th Annual ACM-SIAM Symposium on Discrete Algorithms* (Baltimore, MD). ACM, New York, 885–886.

COWEN, L., AND WAGNER, C. 2000. Compact roundtrip routing in directed graphs. In *Proceedings of the 19th Annual ACM Symposium on Principles of Distributed Computing* (Portland, OR). ACM, New York, 51–59.

FRAIGNIAUD, P., AND GAVOILLE, C. 2001. Routing in trees. In *Proceedings of the 28th International Colloquium on Automata, Languages and Programming* (Crete, Greece), 757–772.

FREDMAN, M., KOMLÓS, J., AND SZEMERÉDI, E. 1984. Storing a sparse table with $O(1)$ worst case access time. *J. ACM 31*, 538–544.

HAGERUP, T., MILTERSEN, P., AND PAGH, R. 2001. Deterministic dictionaries. *J. Algo. 41*, 69–85.

PAGH, R. 2000. Faster deterministic dictionaries. In *Proceedings of the 11th Annual ACM-SIAM Symposium on Discrete Algorithms* (San Francisco, CA). ACM, New York, 487–493.

PELEG, D. 2000. *Distributed Computing — A Locality-Sensitive Approach.* Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA.

THORUP, M., AND ZWICK, U. 2001. Compact routing schemes. In *Porceedings of the 13th Annual ACM Symposium on Parallel Algorithms and Architectures* (Crete, Greece). ACM, New York, 1–10.

THORUP, M., AND ZWICK, U. 2005. Approximate distance oracles. *J. ACM 52,* 1, 1–24.

RECEIVED SEPTEMBER 2006; ACCEPTED MARCH 2007