Compact Routing Schemes in Undirected and Directed Graphs

Avi Kadria
Bar Ilan University, Ramat Gan, Israel

Liam Roditty
Bar Ilan University, Ramat Gan, Israel

Abstract

In this paper, we study the problem of compact routing schemes in weighted undirected and directed graphs.

For weighted undirected graphs, more than a decade ago, Chechik [PODC'13] presented a $\approx 3.68k$-stretch compact routing scheme that uses $\tilde{O}(n^{1/k} \log D)$ local storage, where D is the normalized diameter, for every $k > 1$. We present a $\approx 2.64k$-stretch compact routing scheme that uses $\tilde{O}(n^{1/k})$ local storage *on average* in each vertex. This is the first compact routing scheme that uses total local storage of $\tilde{O}(n^{1+1/k})$ while achieving a $c \cdot k$ stretch, for a constant $c < 3$.

In real-world network protocols, messages are usually transmitted as part of a communication session between two parties. Therefore, more than two decades ago, Thorup and Zwick [SPAA'01] considered compact routing schemes that establish a communication session using a handshake. In their handshake-based compact routing scheme, the handshake is routed along a $(4k-5)$-stretch path, and the rest of the communication session is routed along an optimal $(2k-1)$-stretch path. It is straightforward to improve the $(4k-5)$-stretch of the handshake to $\approx 3.68k$-stretch using the compact routing scheme of Chechik [PODC'13]. We improve the handshake stretch to the optimal $(2k-1)$, by borrowing the concept of roundtrip routing from directed graphs to *undirected* graphs.

For weighted directed graphs, more than two decades ago, Roditty, Thorup, and Zwick [SODA'02 and TALG'08] presented a $(4k+\varepsilon)$-stretch compact roundtrip routing scheme that uses $\tilde{O}(n^{1/k})$ local storage for every $k \geq 3$. For $k = 3$, this gives a $(12+\varepsilon)$-roundtrip stretch using $\tilde{O}(n^{1/3})$ local storage. We improve the stretch by developing a 7-roundtrip stretch routing scheme with $\tilde{O}(n^{1/3})$ local storage. In addition, we consider graphs with bounded hop diameter and present an optimal $(2k-1)$-roundtrip stretch routing scheme that uses $\tilde{O}(D_{HOP} \cdot n^{1/k})$, where D_{HOP} is the hop diameter of the graph.

2012 ACM Subject Classification Theory of computation → Routing and network design problems

Keywords and phrases Routing schemes, Compact routing schemes, Distance oracles, Computer networks, Graph algorithms

Digital Object Identifier 10.4230/LIPIcs.DISC.2025.38

Related Version *Full Version*: `https://arxiv.org/abs/2503.13753` [15]

Funding *Liam Roditty*: Supported in part by BSF grants 2016365 and 2020356.

1 Introduction

Routing is a fundamental task in computer networks. A *routing scheme* is a mechanism designed to deliver messages efficiently from a source vertex to a destination vertex within the network. In this paper, we study both undirected and directed weighted graphs, aiming to route along short paths.


© Avi Kadria and Liam Roditty;
licensed under Creative Commons License CC-BY 4.0

More specifically, a routing scheme is composed of a preprocessing phase and a routing phase. In the preprocessing phase, the entire graph is accessible, allowing the preprocessing algorithm to compute a routing table and a label for each vertex[1], which is then stored in the local storage of each vertex. In the routing phase, the routing algorithm at each vertex on the routing path can only access the local storage of the vertex. The routing algorithm gets as input a message, a destination label, and possibly a header, and decides which of the vertex neighbors is the next vertex on the routing path. The routing continues until the message reaches the destination.

A *compact* routing scheme is a routing scheme that uses $o(n)$ space in the local storage on average at each vertex, where n is the number of vertices in the graph. Let u be a source vertex and let v be a destination vertex. We denote by $\hat{d}(u,v)$ the length of the path used by the routing algorithm to route a message from u to v. The *stretch* of the routing scheme is defined as $\max_{u,v\in V}(\frac{\hat{d}(u,v)}{d(u,v)})$, where $d(u,v)$ is the distance from u to v. The *roundtrip* stretch is defined as $\max_{u,v\in V}(\frac{\hat{d}(u,v)+\hat{d}(v,u)}{d(u,v)+d(v,u)})$.

The design of efficient compact routing schemes in undirected graphs has been a well-studied subject in the last few decades, see for example [19, 3, 4, 8, 11, 24, 22, 7]. In Table 1 we summarize the previous results.

From the Erdős girth conjecture, it follows that every routing scheme with stretch $< 2k+1$ must use a total storage of $\Omega(n^{1+1/k})$ bits. The approximate distance oracle data structure of Thorup and Zwick [25], which is implemented in the centralized model, where all the information is available upon a distance query to the data structure, has an optimal $(2k-1)$-stretch with $\tilde{O}(n^{1+1/k})$ total storage. In light of the gap between routing schemes and approximate distance oracles, the following problem is natural.

▶ **Problem 1.** *For every $k \geq 2$, given a weighted undirected graph, what is the best stretch of a routing scheme that uses $\tilde{O}(n^{1+1/k})$ total storage?*

The $\approx 3.68k$-stretch compact routing scheme of Chechik [7], from more than a decade ago, is the current best stretch with $\tilde{O}(n^{1/k})$ worst-case local storage. In this paper, we improve the stretch to $\approx 2.64k$ by allowing an average local storage of $\tilde{O}(n^{1/k})$, as presented in the following theorem.

▶ **Theorem 1.** *Let $G=(V,E,w)$ be a weighted undirected graph. For every $k \geq 3$, there is an $\approx 2.64k$-stretch compact routing scheme that uses local routing tables of an average size of $\tilde{O}(n^{1/k})$, vertex labels of size $\tilde{O}(k)$ and packet headers of size $\tilde{O}(k)$.*

All the compact routing schemes mentioned so far solve the problem of sending a single message from the source to the destination, while in most real-world network applications, two parties communicate over the network for a session. A *communication session* is composed of two phases. In the first phase, a connection is established between the source and the destination, and in the second phase, a stream of messages is exchanged between the parties. Many real-world protocols, such as `TLS`, `QUIC`, `TCP`, `SSH`, `Wi-Fi`, and `Bluetooth`, adhere to this framework.

We consider the *handshake* mechanism for the establishment phase, as presented by Thorup and Zwick [24]. The handshake is composed of two messages of size $\tilde{O}(1)$, that are exchanged between the parties to establish the connection. The first message is sent from the

[1] In this paper, we study labeled routing schemes, where the preprocessing algorithm can assign labels to the vertices. When the vertex labels are fixed and cannot be changed, the routing scheme is called an name-independent routing scheme (see, for example, [1, 16, 17, 2, 13])

Table 1 Compact routing schemes in undirected graphs.

Stretch	Local storage	Uses average local storage?	Ref.	Comments
$O(k)$	$\tilde{O}(n^{1/k})$	yes	[19]	unweighted graphs
2^k-1	$\tilde{O}(n^{1/k})$	yes	[3]	
$O(k^2)$	$\tilde{O}(n^{1/k})$	no	[4]	
$4k-5$	$\tilde{O}(n^{1/k})$	no	[24]	
$\approx 3.68k$	$\tilde{O}(n^{1/k})$	no	[7]	
$4k-7+\varepsilon$	$\tilde{O}(n^{1/k})$	no	[22]	
$\approx 2.64k$	$\tilde{O}(n^{1/k})$	yes	Theorem 1	
$2k-1$	$\tilde{O}(n^{1/k})$	no	Theorem 2	roundtrip stretch

source to the destination, and the second message is sent from the destination back to the source. Since the handshake is composed of a message sent from the source to the destination and back, the stretch of the handshake is the roundtrip stretch defined earlier.

Thorup and Zwick [24] presented a compact routing scheme that uses a handshake, in which two messages are routed along a $(4k-5)$-stretch path, to establish a connection. Then, a stream of messages is routed along an optimal $(2k-1)$-stretch path. While the compact routing scheme of Chechik [7] achieves an $\approx 3.68k$-roundtrip stretch for the handshake, there is still a significant gap from the optimal $(2k-1)$-stretch followed by the Erdős girth conjecture. Therefore, the main open problem for routing in a communication session is to reduce the stretch of the handshake phase and obtain a compact roundtrip routing scheme with improved stretch.

▶ **Problem 2.** *For every $k \geq 2$, given a weighted undirected graph, what is the best roundtrip stretch of a routing scheme that uses $\tilde{O}(n^{1/k})$ local storage?*

In this paper we *solve* Problem 2 by presenting an *optimal* $(2k-1)$-roundtrip stretch for the handshake phase, that matches the lower bound that follows from the Erdős girth conjecture, as presented in the following theorem.

▶ **Theorem 2.** *Let $G=(V,E,w)$ be a weighted undirected graph. Let $k \geq 1$ be an integer. There is a $(2k-1)$-stretch compact roundtrip routing scheme that uses local routing tables of size $\tilde{O}(n^{1/k})$, vertex labels of size $\tilde{O}(k)$ and packet headers of size $\tilde{O}(k)$.*

Using this result with the handshake-based routing scheme of Thorup and Zwick [24], one obtains an optimal $(2k-1)$-stretch compact routing scheme for any communication session. We summarize our new results for undirected graphs and compare them to the previous work in Table 1.

Next, we turn our attention to weighted *directed* graphs. Since preserving the asymmetric reachability structure of directed graphs requires $\Omega(n^2)$ space[2], no spanner, emulator, or compact routing scheme can exist for directed graphs. Cowen and Wagner [9] circumvented the $\Omega(n^2)$ space lower bound by introducing roundtrip distances in directed graphs, defined as $d(u \leftrightarrow v) = d(u,v) + d(v,u)$.

[2] In a bipartite graph in which all the edges are from one side to the other, there are $\Theta(n^2)$ edges and removing each of them makes the destination not reachable from the source. Thus, storing reachability requires $\Omega(n^2)$ space.

In the last few decades, a few compact roundtrip routing schemes were presented, see for example [9, 10, 21]. The state-of-the-art result was obtained by Roditty, Thorup, and Zwick [21]. They presented a 3-stretch compact roundtrip routing scheme that uses $\tilde{O}(n^{1/2})$ local storage and also a $(4k+\varepsilon)$-stretch compact roundtrip routing scheme that uses $\tilde{O}(n^{1/k})$ local storage for every $k \geq 3$.

From the Erdős girth conjecture, it follows that every compact roundtrip routing scheme with stretch $< 2k+1$ must use total storage of $\Omega(n^{1+1/k})$ bits. Closing the gap between the upper and the lower bound is the main open problem regarding compact roundtrip routing schemes.

▶ **Problem 3.** *For every $k \geq 2$, given a directed weighted graph, what is the best stretch of a roundtrip routing scheme that uses $\tilde{O}(n^{1/k})$ local storage?*

In recent years, roundtrip distances have been extensively studied but only in the context of roundtrip spanners and roundtrip emulators (see, for example [14, 6, 21, 23, 5, 18]). Despite all the recent progress, no improvements were obtained for compact roundtrip routing schemes since the $(4k+\varepsilon)$-stretch roundtrip routing scheme of [21] from more than two decades ago.

In this paper, we improve upon [21] for the case that $k = 3$. More specifically, using $\tilde{O}(n^{1/3})$ local storage, Roditty, Thorup, and Zwick [21] obtained a $(12+\varepsilon)$-stretch roundtrip routing scheme. We present a 7-stretch roundtrip routing scheme that uses $\tilde{O}(n^{1/3})$ local storage, as presented in the following theorem.

▶ **Theorem 3.** *Let $G = (V, E, w)$ be a weighted directed graph. There is a 7-stretch compact roundtrip routing scheme that uses local routing tables of size $\tilde{O}(n^{1/3})$, vertex labels of size $\tilde{O}(1)$ and packet headers of size $\tilde{O}(1)$.*

In addition, in the following theorem, we present an optimal[3], up to polylogarithmic factors, compact roundtrip routing scheme in graphs with $D_{hop} = \tilde{O}(k)$, where D_{hop} is the hop diameter of the graph.

▶ **Theorem 4.** *Let $G = (V, E, w)$ be a weighted directed graph. Let $k \geq 1$ be an integer. There is a $(2k-1)$-stretch compact roundtrip routing scheme that uses local routing tables of size $\tilde{O}(D_{hop}n^{1/k})$, vertex labels of size $\tilde{O}(D_{hop}k)$ and packet headers of size $\tilde{O}(D_{hop})$.*

We summarize our new results for directed graphs and compare them to the previous work in Table 2.

The rest of this paper is organized as follows. In Section 2 we present some necessary preliminaries. In Section 3 we present an overview of our technical contributions and our new compact routing schemes. In Section 4 we present our optimal compact roundtrip routing scheme for weighted undirected graphs. In Section 5 we extend Section 4 to directed graphs with small hop diameter. In Appendix A we present our 7-stretch compact roundtrip routing scheme for weighted directed graphs. Finally, in Section 6 in the full version of this paper [15], we present our single message compact routing scheme for weighted undirected graphs that use average local storage.

[3] Assuming the Erdős conjecture, it is easy to create a graph G with $\Omega(n^{1+1/k})$ edges such that the girth of G is $2k+2$, and the diameter of G is at most $2k+1$. Given a graph with $\Omega(n^{1+1/k})$ edges and $2k+2$ girth, if there is a pair of vertices $u, v \in V$ such that $d(u,v) \geq 2k+2$, we can add an edge between u and v to the graph without reducing the girth. By repeating this process until there are no pairs $u, v \in V$ such that $d(u,v) \geq 2k+2$, we get that the diameter is at most $2k+1$.

Table 2 Compact roundtrip routing results in directed weighted graphs.[4]

Stretch	local storage	Uses average local storage?	Ref.	Comments
3	$\tilde{O}(n^{1/2})$	yes	[9]	
3	$\tilde{O}(n^{2/3})$	no	[9]	
2^k-1	$\tilde{O}(n^{1/k})$	yes	[10]	
3	$\tilde{O}(n^{1/2})$	no	[21]	
$4k+\varepsilon$	$\tilde{O}(\frac{1}{\varepsilon}n^{1/k})$	no	[21]	$\varepsilon > 0$
$12+\varepsilon$	$\tilde{O}(\frac{1}{\varepsilon}n^{1/3})$	no	[21]	$\varepsilon > 0$
7	$\tilde{O}(n^{1/3})$	no	Theorem 3	
$2k-1$	$\tilde{O}(D_{hop} \cdot n^{1/k})$	no	Theorem 4	D_{hop} is the hop diameter of G.

2 Preliminaries

Let $G = (V, E, w)$ be a weighted graph with n vertices and m edges, where $w : E \to \mathbb{R}^+$. We consider both connected undirected graphs and strongly connected directed graphs[5].

Let the distance $d_G(u, v)$ from u to v be the length of the shortest path from u to v in G, where the length of a path is the sum of its edge weights, and let $P_G(u, v)$ be the shortest path from u to v, G is omitted when it is clear from context. The roundtrip distance $d(u \leftrightarrow v)$ is defined as $d(u, v) + d(v, u)$. Throughout this paper, we assume that for any two vertices u and v, there exists a unique shortest path between them. In the case of multiple shortest paths of the same length, we break ties by selecting the path with the lexicographically smallest sequence of vertex identifiers.

Let $X \subseteq V$. The distance $d(u, X)$ from u to X is the distance between u and the closest vertex to u from X, that is, $d(u, X) = \min_{x \in X}(d(u, x))$. Similarly, the roundtrip distance from u to X is defined as $d(u \leftrightarrow X) = \min_{x \in X}(d(u \leftrightarrow x))$. Let $p(u, X) = \arg\min_{x \in X}(d(u \leftrightarrow x))$ (ties are broken in favor of the vertex with a smaller identifier).

Next, following the ideas of Thorup and Zwick [25], we define bunches and clusters. Let $u \in V$ and let $X, Y \subseteq V$. The bunch of u with respect to X and Y is defined as $B(u, X, Y) = \{v \in X \mid d(u \leftrightarrow v) < d(u \leftrightarrow Y)\}$. The ball of u with respect to Y is defined as $B(u, Y) = \{v \in V \mid d(u \leftrightarrow v) < d(u \leftrightarrow Y)\}$ (notice that $B(u, Y) = B(u, V, Y)$). The cluster of u with respect to Y is defined as $C(u, Y) = \{v \in V \mid d(u \leftrightarrow v) < d(v \leftrightarrow Y)\}$.

The starting point in many algorithms, distance oracles, and compact routing schemes, and in particular in Thorup and Zwick [24] routing scheme, is a hierarchy of vertex sets $A_0, A_1, \ldots, A_k$, where $A_0 = V$, $A_k = \emptyset$, $A_{i+1} \subseteq A_i$ and $|A_i| = n^{1-i/k}$ for $0 \leq i \leq k-1$.

For every $0 \leq i \leq k-1$, the i-th pivot of u is defined as $p_i(u) = p(u, A_i)$, and $h_i(u)$ is defined as $d(u, A_i)$. The i-th bunch of u is defined as $B_i(u) = B(u, A_i, A_{i+1})$. The bunch of u is defined as the union of its individual bunches, that is, $B(u) = \bigcup_{i=0}^{k-1} B_i(u)$. The cluster of a vertex $w \in A_i \setminus A_{i+1}$ is defined as the cluster of w with respect to the set A_{i+1}, that is, $C(w) = C(w, A_{i+1})$. We denote by $[k]$ the set $\{0, 1, 2, \ldots, k-1\}$.

In the following lemma, we provide an upper bound for the size of $B(u)$, which is $O(kn^{1/k})$, as demonstrated by Thorup and Zwick [25].

[4] poly-logarithmic factors are omitted.

[5] If the graph is not connected (or not strongly connected), we add a dummy vertex and connect it with bi-directional edges of weight ∞ to every vertex of the graph.

▶ **Lemma 5** ([25, 20]). *Given an integer parameter $k \geq 2$, we can compute sets $A_1, \ldots, A_{k-1}$, such that $|A_i| = O(n^{1-i/k})$, for every $1 \leq i \leq k-1$. For every $i \in [k]$ the size of $B_i(u)$ is $\tilde{O}(n^{1/k})$.*

In the following lemma, we provide an upper bound for the size of $C(u)$, which is $O(n^{1/k})$, as demonstrated by Thorup and Zwick [24].

▶ **Lemma 6** ([24]). *Given a parameter p, we can compute a set A of size $\tilde{O}(np)$ such that, $|C(w, A)| = O(1/p)$, for every vertex $w \in V \setminus A$, and $|B(v, V, A)| = O(1/p)$ for every $v \in V$.*

Let $S \subseteq V$. We define $T_{out}(u, S)$ as a tree containing the directed shortest paths from u to all the vertices in S, and $T_{in}(u, S)$ as a tree containing the directed shortest paths from all the vertices in S to u. When u is clear from context, we omit it, for example, we use $T(C(u)) = T(u, C(u))$, and $T(B(u)) = T(u, B(u))$. Note that it is possible for $|T_{out}(u, S)|$ to be bigger than $|S|$ in cases where the shortest path from u to a vertex in S passes through a vertex not in S. [6] Let $T(u, X) = T_{in}(u, X) \cup T_{out}(u, X)$, as defined in [21], when u is clear from context we omit it and write $T(X)$. Notice that in undirected graphs, since $T_{in}(u, X) = T_{out}(u, X)$, we have that $T(u, X) = T_{in}(u, X) = T_{out}(u, X)$. Next, we show that if S is a ball, i.e. $S = B(u, X) = B(u, V, X)$ for some set X, then $|T_{out}(u, S)| = |S|$ and $|T_{in}(u, S)| = |S|$, and therefore $|T(u, S)| \leq 2|S|$.

▶ **Lemma 7** ([21]). *$|T_{out}(u, B(u, X))| = |B(u, X)|$ and $|T_{in}(u, B(u, X))| = |B(u, X)|$.*

Proof. Let $u \in V$, $v \in B(u, X)$, and let $w \in P(u, v)$. We will show that $w \in B(u, X)$. Since all the vertices in T_{out} are on the shortest path from u to a vertex in $B(u, X)$, we obtain that $|T_{out}(u, B(u, X))| \leq |B(u, X)|$, as required. The proof for the in-ball is identical for reversed paths. From the triangle inequality, we know that

$$d(u \leftrightarrow w) = d(u, w) + d(w, u) \overset{\triangle}{\leq} d(u, w) + d(w, v) + d(v, u) = d(u \leftrightarrow v) < d(u \leftrightarrow X),$$

where the last inequality follows from the fact that $v \in B(u, X)$.[7] Since $d(u \leftrightarrow w) < d(u \leftrightarrow X)$ we get that $w \in B(u, X)$, as required. ◀

The following lemma was originally proven in [25] for any metric space and therefore holds also for roundtrip distances. For completeness, we prove the lemma for roundtrip distances.

▶ **Lemma 8.** *Let $u, v \in V$ and let $0 < i \leq k-1$. If $p_j(u) \notin B(v)$ and $p_j(v) \notin B(u)$ for every $0 < j < i$, then*

$$d(u \leftrightarrow p_i(u)) \leq i \cdot d(u \leftrightarrow v) \quad \text{and} \quad d(v \leftrightarrow p_i(v)) \leq i \cdot d(u \leftrightarrow v).$$

Proof. We prove the claim by induction for every $0 \leq i \leq \ell$. For $i = 0$, the lemma holds since $d(v \leftrightarrow v) = 0$ and $d(u \leftrightarrow u) = 0$. Next, we prove the induction step. We assume the correctness of the claim for $i-1$ and show its correctness for i. Therefore, $d(v \leftrightarrow p_{i-1}(v)) \leq (i-1) \cdot d(u \leftrightarrow v)$ and $d(u \leftrightarrow p_{i-1}(u)) \leq (i-1) \cdot d(u \leftrightarrow v)$. Without loss of generality, we show that $d(u \leftrightarrow p_i(u)) \leq i \cdot d(u \leftrightarrow v)$. The proof for v is identical.

[6] We denote with $|H|$ the number of edges in H, i.e. $|H| = |E(H)|$.

[7] $\overset{\triangle}{\leq}$ denotes an inequality that follows from the triangle inequality.

Since $i \leq \ell$, it follows that $i-1 < \ell$. Therefore, from the lemma's assumptions, we know that $p_{i-1}(v) \notin B(u)$. From the definition of $B(u)$, it follows that $d(u \leftrightarrow p_i(u)) \leq d(u \leftrightarrow p_{i-1}(v))$. From the triangle inequality, it follows that $d(u \leftrightarrow p_{i-1}(v)) \leq d(u \leftrightarrow v) + d(v \leftrightarrow p_{i-1}(v))$. Recall that from the induction assumption we know that $d(v \leftrightarrow p_{i-1}(v)) \leq (i-1) \cdot d(u \leftrightarrow v)$. Therefore, we get that:

$$\begin{aligned} d(u \leftrightarrow p_i(u)) &\leq d(u \leftrightarrow p_{i-1}(v)) \leq d(u \leftrightarrow v) + d(v \leftrightarrow p_{i-1}(v)) \\ &\leq d(u \leftrightarrow v) + (i-1) \cdot d(u \leftrightarrow v) = i \cdot d(u \leftrightarrow v), \end{aligned}$$

as required. ◀

Next, we show that if $p_{i-1}(v) \notin B(u)$, then $d(v \leftrightarrow p_i(v)) \leq d(v \leftrightarrow p_{i-1}(v)) + 2 \cdot d(u \leftrightarrow v)$.

▶ **Lemma 9.** *Let $u, v \in V$, if $p_{i-1}(v) \notin B(u)$ then $d(v \leftrightarrow p_i(v)) \leq d(v \leftrightarrow p_{i-1}(v)) + 2d(u \leftrightarrow v)$*

Proof. From the definition of $p_i(v)$, we know that it is the closest vertex to v in A_i, and since $p_i(u) \in A_i$ we get that $d(v \leftrightarrow p_i(v)) \leq d(v \leftrightarrow p_i(u))$. From the triangle inequality, it follows that $d(v \leftrightarrow p_i(u)) \leq d(v \leftrightarrow u) + d(u \leftrightarrow p_i(u))$. From the lemma assumption, we know that $p_{i-1}(v) \notin B(u)$. Therefore, $d(u \leftrightarrow p_i(u)) \leq d(u \leftrightarrow p_{i-1}(v))$. From the triangle inequality, it follows that $d(u \leftrightarrow p_{i-1}(v)) \leq d(u \leftrightarrow v) + d(v \leftrightarrow p_{i-1}(v))$. Overall, we get that:

$$\begin{aligned} d(v \leftrightarrow p_i(v)) &\leq d(v \leftrightarrow p_i(u)) \leq d(v \leftrightarrow u) + d(u \leftrightarrow p_i(u)) \leq d(v \leftrightarrow u) + d(u \leftrightarrow p_{i-1}(v)) \\ &\leq d(v \leftrightarrow u) + d(u \leftrightarrow v) + d(v \leftrightarrow p_{i-1}(v)) = 2d(u \leftrightarrow v) + d(v, p_{i-1}(v)), \end{aligned}$$

as required. ◀

The following lemma holds only for *undirected* graphs and is presented in [25].

▶ **Lemma 10** ([25])**.** *Let $G = (V, E, w)$ be a weighted undirected graph. Let $v \in A_i \setminus A_{i+1}$, let $u \in C(v)$, and let $w \in P(v, u)$ then $w \in C(v)$.*

Proof. For the sake of contradiction, assume that $w \notin C(v)$. From the definition of $C(v)$, we have that $d(v \leftrightarrow w) \geq d(w \leftrightarrow A_{i+1})$.

By the triangle inequality, we have: $d(u \leftrightarrow A_{i+1}) \leq d(u \leftrightarrow w) + d(w \leftrightarrow A_{i+1})$. Since $d(w \leftrightarrow A_{i+1}) \leq d(v \leftrightarrow w)$, we get $d(u \leftrightarrow A_{i+1}) \leq d(u \leftrightarrow w) + d(w \leftrightarrow A_{i+1}) \leq d(u \leftrightarrow w) + d(v \leftrightarrow w)$. Since $w \in P(v, u)$, and the graph is undirected, it follows that $d(u \leftrightarrow w) + d(v \leftrightarrow w) = d(v \leftrightarrow u)$. Thus, $d(u \leftrightarrow A_{i+1}) \leq d(v \leftrightarrow u)$, a contradiction to the fact that $u \in C(v)$. ◀

2.1 General framework

A routing scheme comprises two phases: a preprocessing phase and a routing phase. In the preprocessing phase, the entire graph is accessible to the algorithm. The preprocessing algorithm computes for every $u \in V$ a routing table $\mathtt{RT}(u)$ and a label $\mathtt{L}(u)$. Each vertex u saves $\mathtt{RT}(u)$ and $\mathtt{L}(u)$ in its local storage. A routing scheme is considered *compact* if the size of the routing tables is sub-linear in the number of vertices, i.e., $|\mathtt{RT}(u)| = o(n)$.

In the routing phase, the goal is to route a message from a source vertex u to a destination vertex v. Specifically, the routing algorithm at the source vertex u gets as input the routing table $\mathtt{RT}(u)$, and the labels $\mathtt{L}(u)$ and $\mathtt{L}(v)$. Based on this input, the routing algorithm determines a neighboring vertex of u to which the message should be forwarded. The routing algorithm can also attach a header to the message. When a vertex w receives a message, the routing algorithm at w gets as input the routing table $\mathtt{RT}(w)$, and the labels $\mathtt{L}(w)$ and $\mathtt{L}(v)$ (and possibly a header). Based on this input, the routing algorithm determines a neighboring

vertex of w to which the message should be forwarded. The message is routed from a vertex to one of its neighbors until the message reaches its destination vertex v.

We denote by $\hat{d}(u,v)$ the distance that a message whose source vertex is u and whose destination vertex is v traverses from u to v. The stretch of the routing scheme is defined as $\max_{u,v\in V}(\frac{\hat{d}(u,v)}{d(u,v)})$.

Several variants of compact routing schemes exist. In a *labeled* routing scheme, the preprocessing algorithm can assign labels to the vertices. In a *fixed-port* routing scheme, the order of the neighbors of each vertex is fixed and cannot be changed by the preprocessing algorithm. In this work, we focus on labeled, fixed-port compact routing schemes.

2.2 Routing in trees

An essential ingredient in our compact routing schemes for general graphs is the following compact routing scheme for trees. Given a tree T, the preprocessing algorithm assigns a label $\mathtt{L}(v,T)$ to every vertex v in T. The routing algorithm then routes a message from a source vertex u to a destination vertex v on the shortest path from u to v in T. Thorup and Zwick [24] presented a tree routing scheme that uses only vertex labels of size $(1+o(1))\log n$ and no routing tables. In the fixed-port model, however, the label size increases to $O(\log^2 n)$. A similar scheme was presented by Fraigniaud and Gavoille [12]. The following lemma outlines the known results for tree compact routing schemes.

▶ **Lemma 11** ([12, 24]). *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.*

2.3 Routing in directed graphs

Roditty, Thorup, and Zwick [21] extended the compact routing scheme from trees to double trees to handle directed graphs. In our work on directed graphs (Section 5 and Appendix A), we employ this double-tree adaptation when routing within double-trees. Moreover, for the routing in clusters to work in directed graphs, they adjusted the definition of cluster adaptation using the following definition of roundtrip ordering.

▶ **Definition 12.** *We assume that $V=\{1,\ldots,n\}$. Let $u_1,u_2,v\in V$. We say that $u_1\prec_v u_2$ if one of the following holds:*

- $d(v\leftrightarrow u_1)<d(v\leftrightarrow u_1)$
- $d(v\leftrightarrow u_1)=d(v\leftrightarrow u_1)$ *and* $d(u_1\to v)<d(u_1\to v)$
- $d(v\leftrightarrow u_1)=d(v\leftrightarrow u_1)$ *and* $d(u_1\to v)=d(u_1\to v)$ *and* $u_1<u_2$

Using this definition, the cluster of u with respect to Y is defined as $C(u,Y)=\{v\in V\mid u\not\prec_v p_Y(u)\}$, where $p_Y(u)$ satisfies that $p_Y(u)\preceq_v w$ for every $w\in Y$.

Using this definition, they proved the following lemma:

▶ **Lemma 13** ([21]). *If $v\in C(u)$ and $w\in P(u,v)$, then $v\in C(w)$*

3 Overview

In this section, we present an overview of our new compact routing schemes and their main technical contributions. Throughout the overview, let $V=A_0,\ldots,A_k=\emptyset$ be a hierarchy such that, $|A_i|=\tilde{O}(n^{1-i/k})$ and $|B(u)|=\tilde{O}(n^{1/k})$, for every $u\in V$, created using Lemma 5. We denote with $x\to y$ routing from x to y on $P(x,y)$.

Roundtrip routing scheme in undirected graphs. Notice that any t-stretch compact routing scheme is also a t-roundtrip stretch compact routing scheme. In undirected graphs, where $d(u,v) = d(v,u)$, we have $d(u \leftrightarrow v) = d(u,v) + d(v,u) = 2d(u,v)$. This might lead one to question the potential benefits of considering roundtrip routing in undirected graphs. In other words, why might the problem of roundtrip routing be easier than the problem of single message routing, even though $d(u \leftrightarrow v) = 2d(u,v)$?

During the routing process, the information available to the routing algorithm at u may differ from the information available at v. Therefore, the routing path from u to v may differ from the routing path from v to u, which might lead to the case that $\hat{d}(u,v) \neq \hat{d}(v,u)$. Consider for example the case that $\hat{d}(u,v) = 3d(u,v)$ and $\hat{d}(v,u) = 27d(u,v)$. In this case, the roundtrip stretch is 15 while the single message stretch is 27, and even though $d(u \leftrightarrow v) = 2d(u,v)$, the roundtrip stretch is much smaller than the single message stretch.

Next, we provide an overview of our optimal $(2k-1)$-roundtrip stretch compact routing scheme (for the complete description, see Section 4). The preprocessing algorithm sets $\texttt{RT}(u) = \{\texttt{L}(u, T(C(v))) \mid v \in B(u)\}$ and $\texttt{L}(u) = \{\texttt{L}(u, T(C(p_i(u)))) \mid i \in [k]\}$, for every $u \in V$. Let $\ell(x,y) = \min\{i \mid p_i(y) \in B(x)\}$, and let $b = \ell(u,v)$, and let $a = \ell(v,u)$. The roundtrip routing path is $u \to p_b(v) \to v \to p_a(u) \to u$ (see Figure 1).

Our main technical contribution is in Lemma 16, where we show that while the path $u \to p_b(v) \to v$ might be of length at most $(4k-3)d(u,v)$, the entire path $u \to p_b(v) \to v \to p_a(u) \to u$ is of length of at most $(4k-2)d(u,v) = (2k-1)d(u \leftrightarrow v)$, and therefore the roundtrip stretch is the optimal $(2k-1)$-stretch.

Next, we provide some intuition why $\hat{d}(u \leftrightarrow v) \leq (2k-1)d(u \leftrightarrow v)$. From the triangle inequality, we have that $\hat{d}(u \leftrightarrow v) \leq d(u \leftrightarrow v) + d(u \leftrightarrow p_a(u)) + d(v \leftrightarrow p_b(v))$. From the definition of a and b, for every $0 < i < a \leq b \leq k-1$ we have $p_i(u) \notin B(v)$ and $p_i(v) \notin B(u)$, and for every $a \leq i < b$ we have $p_i(v) \notin B(u)$. This allows us to prove that $d(u \leftrightarrow p_a(u)) \leq a \cdot d(u \leftrightarrow v)$ and $d(v \leftrightarrow p_b(v)) \leq a \cdot d(u \leftrightarrow v) + (b-a) \cdot 2d(v \leftrightarrow p_b(v))$. Overall:

$$\begin{aligned} d(u \leftrightarrow p_a(u)) + d(v \leftrightarrow p_b(v)) &\leq ad(u \leftrightarrow v) + ad(u \leftrightarrow v) + (b-a)2d(u \leftrightarrow v) \\ &= 2bd(u \leftrightarrow v) \leq 2(k-1)d(u \leftrightarrow v) \end{aligned}$$

and therefore $\hat{d}(u \leftrightarrow v) \leq d(u \leftrightarrow v) + d(u \leftrightarrow p_a(u)) + d(v \leftrightarrow p_b(v)) \leq (2k-1)d(u \leftrightarrow v)$.

Directed roundtrip routing schemes. One might wonder why the general compact roundtrip routing scheme presented above for undirected graphs can not be extended to directed graphs. The problem lies in the structure of clusters. In undirected graphs, if $u \in C(w)$ then $P(u,w) \subseteq C(w)$ (see Lemma 10), and therefore we can route from u to every w, such that $u \in C(w)$. Unfortunately, in directed graphs, this nice property of clusters does not necessarily hold. More specifically, it might be that $P(u,w) \not\subseteq C(w)$ even when $u \in C(w)$ and as a result, we cannot route from u to w while using only the cluster $C(w)$ as in undirected graphs. A simple approach to overcome this problem is to store $P(u,w)$, for every $w \in B(u)$, in $\texttt{RT}(u)$. Using this approach, we can extend the roundtrip routing scheme from undirected graphs to directed graphs with small hop diameter (see Theorem 4).

For routing tables of size $\tilde{O}(n^{1/3})$, we overcome the problem that $P(u,w) \not\subseteq C(w)$ using a more sophisticated solution. For an hierarchy with $k = 3$ we have $V = A_0, A_1, A_2, A_3 = \emptyset$ and three bunches, $B_0(u), B_1(u)$ and $B_2(u)$, for every $u \in V$. Let $u, v \in V$. To follow the compact roundtrip of undirected graphs we need to be able to route from u to v, if $v \in B_0(u)$, from u to $p_1(v)$, if $p_1(v) \in B_1(u)$, and from u to $p_2(v)$, otherwise. Since $B_0(u)$ is a ball

$P(u,v) \subseteq B_0(u)$, for every $v \in B_0(u)$ (see Lemma 7). Therefore, we can route from u to every vertex in $B_0(u)$. Moreover, since $C(p_2(v)) = V$, we can route from u to $p_2(v)$. This leaves us with the challenge of routing from u to $p_1(v)$ when $p_1(v) \in B_1(u)$.

To handle this challenge, we divide the routing from u to $p_1(v)$ into two cases. If $P(u,p_1(v)) \subseteq C(p_1(v))$, we simply route on the path $P(u,p_1(v))$. Otherwise, if $P(u,p_1(v)) \not\subseteq C(p_1(v))$, we let $z \in P(u,p_1(v))$ be the first vertex such that $z \notin C(p_1(v))$. In this case, we route along the path $u \to p_2(z) \to p_1(v)$. Using the fact that $z \notin C(p_1(v))$ we show that $d(u,p_2(z)) + d(p_2(z),p_1(v)) \leq d(u,p_1(v)) + d(u \leftrightarrow p_1(v))$, and that $\hat{d}(u \leftrightarrow v) \leq 7d(u \leftrightarrow v)$ (see Lemma 19).

Average single message routing schemes in undirected graphs. Recall that $h_i(u) = d(u,A_i)$. In the preprocessing algorithm, we set $\texttt{RT}(u) = \{\texttt{L}(u,T(C(v))) \mid v \in B(u)\} \cup \{L(v,T(C(u))) \mid v \in C(u)\}$, and $\texttt{L}(u) = \{\langle h_i(u), \texttt{L}(u,T(C(p_i(u))))\rangle \mid i \in [k]\}$, for every $u \in V$.

For the sake of simplicity, we assume that when routing from the source vertex u to the destination vertex v, the value $d(u,v)$ is known to the routing algorithm. In this case, we present an optimal $(2k-1)$-stretch routing scheme in Section 6 of the full version of this paper [15]. The algorithm at the source u works as follows. For each $0 \leq i \leq k-1$, if $p_i(v) \in B_i(u)$, it routes from u to v in $T(C(p_i(v)))$. Otherwise, if the inequality $h_{i+1}(v) > d(u,v) + h_i(u)$ holds, then we have that $v \in C(p_i(u))$ and therefore the algorithm routes from u to v in $T(C(p_i(u)))$. If neither condition is satisfied, the algorithm proceeds to the next iteration. The $(2k-1)$-stretch of the routing scheme follows by induction using standard tools (see the full version[15] for the complete proof).

In Section 6 of the full version[15], we present a routing algorithm that routes from u to v without knowing the value of $d(u,v)$. To achieve this, we introduce an estimate $\hat{\delta}$ satisfying $\hat{\delta} \leq d(u,v)$, which serves as our current best lower bound for $d(u,v)$. Note that since $\hat{\delta}$ is only an estimate and may be strictly smaller than $d(u,v)$, it is possible that $h_{2i+1}(v) > \hat{\delta} + h_{2i}(u)$ and $v \notin C(p_{2i}(u))$.[8] Therefore, we introduce a condition that, if satisfied, means that $\hat{\delta} \ll h_{2i+1}(v) - h_{2i}(u)$. If the condition is satisfied, the routing algorithm routes to $p_{2i}(u)$ and checks in $\texttt{RT}(p_{2i}(u))$ whether $v \in C(p_{2i}(u))$. If $v \in C(p_{2i}(u))$ the algorithm simply routes from $p_{2i}(u)$ to v. Otherwise, if $v \notin C(p_{2i}(u))$, then we know that $h_{2i+1}(v) - h_{2i}(u) \leq d(u,v)$, and we can safely update $\hat{\delta}$ to $h_{2i+1}(v) - h_{2i}(u)$. We proceed by routing back from $p_{2i}(u)$ to u and then continuing to the next iteration of the algorithm.

The routing algorithm is simple and works as follows. Let $\ell = \min\{i \mid p_i(v) \in B(u)\}$, and let $\hat{\delta} = \max_{0 \leq i \leq \ell}(h_{i+1}(u) - h_i(v))$. For each $0 \leq i \leq \ell/2$, let $1 < c_i < 2$ be a constant:

1. If $h_{2i+1}(v) \leq c_i \cdot \hat{\delta} + h_{2i}(u)$, then the algorithm continues to the next iteration.
2. Otherwise, if $h_{2i+1}(v) > c_i \cdot \hat{\delta} + h_{2i}(u)$, then the algorithm routes to $p_{2i}(u)$ and accesses $\texttt{RT}(p_{2i}(u))$:
   a. If $v \in C(p_{2i}(u))$ then the algorithm routes directly from $p_{2i}(u)$ to v in $T(C(p_{2i}(u)))$.
   b. Otherwise, if $v \notin C(p_{2i}(u))$, the algorithm sets $\hat{\delta}$ to $h_{2i+1}(v) - h_{2i}(u)$ and routes back to u from $p_{2i}(u)$. The algorithm then continues to the next iteration.

In contrast to the simplicity of the routing algorithm, analyzing its stretch is rather involved. For a complete proof, see Section 6 of the full version[15].

[8] We alternate between bounding $h_i(u)$ and $h_i(v)$, but by the definition of $\hat{\delta}$, we have that $h_{i+1}(u) \leq \hat{\delta} + h_i(v)$ for every $i \leq \ell$. Therefore, it suffices to only bound the $h_{2i+1}(v) - h_{2i}(u)$ for $0 \leq i \leq a$.

4 Optimal roundtrip routing in undirected graphs

In this section, we consider roundtrip routing in weighted undirected graphs. In undirected graphs $d(u,v) = d(v,u)$, the roundtrip distance $d(u \leftrightarrow v)$ is simply $d(u,v)+d(v,u) = 2d(u,v)$.

This observation naturally leads to the question: what are the potential advantages of studying *roundtrip* routing in undirected graphs? In particular, why might roundtrip routing be easier to approximate than single-message routing, despite the fact that the roundtrip distance is always twice the one-way distance, i.e., $d(u \leftrightarrow v) = 2d(u,v)$?

The key distinction lies in the asymmetry of available information during the routing process. When routing from a source vertex u to a destination vertex v, the algorithm has access only to the routing table $\texttt{RT}(u)$ and the label $\texttt{L}(v)$. However, routing from v to u is based solely on $\texttt{RT}(v)$ and $\texttt{L}(u)$. Since these inputs may differ significantly, the resulting routing paths may differ significantly, and we may have that $\hat{d}(u,v) \neq \hat{d}(v,u)$.

In single-message routing, the stretch must hold for the worst-case direction. Therefore, $\frac{\max\left(\hat{d}(u,v),\hat{d}(v,u)\right)}{d(u,v)}$ is bounded. However, roundtrip routing requires only that the average of the two directions be bounded. Therefore, $\frac{\hat{d}(u\leftrightarrow v)}{d(u\leftrightarrow v)} = \frac{\hat{d}(u,v)+\hat{d}(v,u)}{2d(u,v)}$ is bounded.

This relaxation in the approximation requirement allows us to achieve an optimal stretch compact roundtrip routing scheme, as presented in the following theorem.

▶ **Reminder of Theorem 2.** *Let $G = (V,E,w)$ be a weighted undirected graph. Let $k \geq 1$ be an integer. There is a $(2k-1)$-stretch compact roundtrip routing scheme that uses local routing tables of size $\tilde{O}(n^{1/k})$, vertex labels of size $\tilde{O}(k)$ and packet headers of size $\tilde{O}(k)$.*

First, we describe the preprocessing algorithm, which computes the routing tables and assigns vertex labels. The input to the algorithm is a graph $G = (V,E,w)$ and an integer parameter $k > 1$. The algorithm uses Lemma 5 to build a hierarchy of vertex sets $A_0, A_1, \ldots, A_k$, where $A_0 = V$, $A_k = \emptyset$, $A_{i+1} \subseteq A_i$, $|A_i| = n^{1-i/k}$ and $|B_i(u)| = \tilde{O}(n^{1/k})$, for every $0 \leq i \leq k-1$. Next, for every $u \in V$, the preprocessing algorithm computes $B(u)$ and $C(u)$. Then, for every $u \in V$, the algorithm sets the routing table $\texttt{RT}(u)$ to:

$$\texttt{RT}(u) = \{\texttt{L}(u, T(C(v))) \mid v \in B(u)\},$$

and the label $\texttt{L}(u)$ to:

$$\texttt{L}(u) = \{\texttt{L}(u, T(C(p_i(u)))) \mid i \in [k]\}.$$

We now turn to bound the size of the routing tables and the vertex labels.

▶ **Lemma 14.** *$|\texttt{RT}(u)| = \tilde{O}(n^{1/k})$, and $|\texttt{L}(u)| = \tilde{O}(k)$, for every $u \in V$.*

Proof. From Lemma 5 it follows that $|B(u)| = \tilde{O}(n^{1/k})$. From Lemma 11 it follows that for every tree T it holds that $|\texttt{L}(u,T)| = \tilde{O}(1)$. Therefore, $|\texttt{RT}(u)| = |B(u)| \cdot \tilde{O}(1) = \tilde{O}(n^{1/k})$, as required. The label of each vertex is composed of k tree labels $L(u, T(C(p_i(u))))$ for every $i \in [k]$. From Lemma 11 we know that $|L(u, T(C(p_i(u))))| = \tilde{O}(1)$. Therefore, $|\texttt{L}(u)| = k \cdot \tilde{O}(1) = \tilde{O}(k)$, as required. ◀

The routing algorithm routes a message from u to v as follows. Let $\ell(x,y) = \min\{i \mid p_i(y) \in B(x)\}$ and let $\ell = \ell(u,v)$. We route from u to v on the shortest path between u and v in $T(C(p_\ell(v)))$. In figure 1 we show the roundtrip route between u and v according to this routing algorithm.

Next, we show that for every w on the shortest path from u to v in $T(C(p_\ell(v)))$ we have that $\texttt{L}(w, T(C(p_\ell(v)))) \in \texttt{RT}(w)$ and therefore w can route to v in $T(C(p_\ell(v)))$.

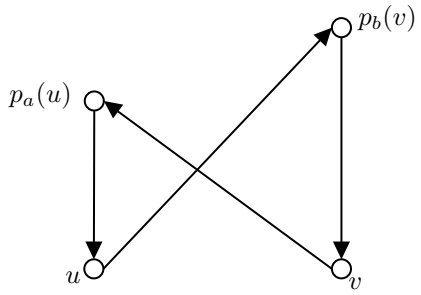


Figure 1 The roundtrip routing of Theorem 2. Let $a = \ell(v,u), b = \ell(u,v)$, where $\ell(x,y) = \min\{i \mid p_i(y) \in B(x)\}$. (We assume wlog that $b \geq a$.).

▶ **Lemma 15.** *For every $w \in P(u, p_\ell(v)) \cup P(p_\ell(v), v)$ it holds that $\mathtt{L}(w, T(C(p_\ell(v)))) \in \mathtt{RT}(w)$*

Proof. If $w \in P(u, p_\ell(v))$, then by Lemma 10, we know that $w \in C(p_\ell(v))$. Similarly, if $w \in P(p_\ell(v), v)$, then from Lemma 10, we know that $w \in C(p_\ell(v))$. Since $w \in C(p_\ell(v))$ it follows from the definition of $C(p_\ell(v))$ that $p_\ell(v) \in B(w)$. By the definition of $\mathtt{RT}(w)$ since $p_\ell(v) \in B(w)$ it follows that $\mathtt{L}(w, T(C(p_\ell(v)))) \in \mathtt{RT}(w)$, as required. ◀

We now turn to the main technical contribution of this section and prove that the stretch of the compact roundtrip routing scheme is $2k-1$.

▶ **Lemma 16.** $\hat{d}(u \leftrightarrow v) \leq (2k-1) \cdot d(u \leftrightarrow v)$

Proof. Let $u, v \in V$, let $b = \ell(u,v) = \min\{i \in [k] \mid p_i(v) \in B(u)\}$, and let $a = \ell(v,u) = \min\{i \in [k] \mid p_i(u) \in B(v)\}$. Assume, without loss of generality, that $b \geq a$. See Figure 1 for an illustration.

In the routing phase, we route from u to v on the shortest path between u and v in $T(C(p_b(v)))$. Similarly, we route from v to u on the shortest path between v and u in $T(C(p_a(u)))$. Therefore,

$$\begin{aligned}\hat{d}(u,v) &= d_{T(C(p_b(v)))}(u,v) \leq d(u,p_b(v)) + d(p_b(v),v) \text{ and } \hat{d}(v,u)\\ &= d_{T(C(p_a(u)))}(u,v) \leq d(v,p_a(u)) + d(p_a(u),u).\end{aligned}$$

From the triangle inequality, we have $d(u,p_b(v)) \leq d(u,v) + d(v,p_b(v))$, therefore we get that:

$$\hat{d}(u,v) = d(u,p_b(v)) + d(p_b(v),v) \leq d(u,v) + d(v,p_b(v)) + d(p_b(v),v) = d(u,v) + d(v \leftrightarrow p_b(v))$$

By symmetry, we also get that $\hat{d}(v,u) \leq d(v,u) + d(u \leftrightarrow p_a(u))$. By definition $\hat{d}(u \leftrightarrow v) = \hat{d}(u,v) + \hat{d}(v,u)$. Therefore, we get:

$$\begin{aligned}\hat{d}(u \leftrightarrow v) &= \hat{d}(u,v) + \hat{d}(v,u)\\ &\leq d(u,v) + d(v \leftrightarrow p_b(v)) + d(v,u) + d(u \leftrightarrow p_a(u))\\ &= d(u \leftrightarrow v) + d(v \leftrightarrow p_b(v)) + d(u \leftrightarrow p_a(u)).\end{aligned}$$

To get that $\hat{d}(u \leftrightarrow v) \leq (2k-1)d(u \leftrightarrow v)$, we show that $d(v \leftrightarrow p_b(v)) + d(u \leftrightarrow p_a(u)) \leq 2(k-1)d(u \leftrightarrow v)$ in the following claim.

▷ Claim 16.1. $d(v \leftrightarrow p_b(v)) + d(u \leftrightarrow p_a(u)) \leq 2(k-1)d(u \leftrightarrow v)$

Proof. Let $\Delta_i^v = d(v \leftrightarrow p_i(v)) - d(v \leftrightarrow p_{i-1}(v))$, for every $0 < i < k$. Notice that for every $0 < j < k$, it holds that:

$$\sum_{i=1}^{j} \Delta_i^v = d(v \leftrightarrow p_j(v)) - d(v \leftrightarrow p_{j-1}(v)) + d(v \leftrightarrow p_{j-1}(v)) - \cdots + d(v \leftrightarrow p_1(v)) - d(v \leftrightarrow p_0(v))$$
$$= d(v \leftrightarrow p_j(v)).$$

Since we assume (wlog) that $b \geq a$, we have:

$$d(v \leftrightarrow p_b(v)) = \sum_{i=1}^{b} \Delta_i^v = \sum_{i=1}^{a} \Delta_i^v + \sum_{i=a+1}^{b} \Delta_i^v = d(v \leftrightarrow p_a(v)) + \sum_{i=a+1}^{b} \Delta_i^v. \tag{1}$$

From the definition of b and a, for every $0 < i < a \leq b$, we have that both $p_i(u) \notin B(v)$ and $p_i(v) \notin B(u)$. Therefore, by applying Lemma 8 we get:

$$d(u \leftrightarrow p_a(u)) \leq a \cdot d(u \leftrightarrow v) \quad \text{and} \quad d(v \leftrightarrow p_a(v)) \leq a \cdot d(u \leftrightarrow v). \tag{2}$$

For every $a \leq i < b$, since $p_i(v) \notin B(u)$, we can apply Lemma 9 to get that $\Delta_i^v < 2d(u \leftrightarrow v)$. Therefore, we get:

$$\sum_{i=a+1}^{b} \Delta_i^v \leq \sum_{i=a+1}^{b} 2d(u \leftrightarrow v) = 2(b-a)d(u \leftrightarrow v) \tag{3}$$

Using the above three inequalities, we get:

$$\begin{aligned} d(u \leftrightarrow p_a(u)) + d(v \leftrightarrow p_b(v)) &\overset{(1)}{=} d(u \leftrightarrow p_a(u)) + d(v \leftrightarrow p_a(v)) + \sum_{i=a+1}^{b} \Delta_i^v \\ &\overset{(2)}{\leq} a \cdot d(u \leftrightarrow v) + a \cdot d(u \leftrightarrow v) + \sum_{i=a+1}^{b} \Delta_i^v \\ &\overset{(3)}{\leq} a \cdot d(u \leftrightarrow v) + a \cdot d(u \leftrightarrow v) + (b-a) \cdot 2d(u \leftrightarrow v) \\ &= 2bd(u \leftrightarrow v) \leq 2(k-1)d(u \leftrightarrow v), \end{aligned}$$

where the last inequality follows from the fact that $b \leq k-1$. ◁

Finally, since we know that $\hat{d}(u \leftrightarrow v) \leq d(u \leftrightarrow v) + d(v \leftrightarrow p_b(v)) + d(u \leftrightarrow p_a(u))$, and from Claim 16.1 we have that $d(v \leftrightarrow p_b(v)) + d(u \leftrightarrow p_a(u)) \leq 2(k-1)d(u \leftrightarrow v)$ we get:

$$\hat{d}(u \leftrightarrow v) \leq d(u \leftrightarrow v) + d(v \leftrightarrow p_b(v)) + d(u \leftrightarrow p_a(u))$$
$$\leq d(u \leftrightarrow v) + 2(k-1)d(u \leftrightarrow v) = (2k-1)d(u \leftrightarrow v),$$

As required. ◀

Theorem 2 follows from Lemma 14, Lemma 15 and Lemma 16.

5 Extending Section 4 to directed graphs

In this section, we consider roundtrip routing in weighted directed graphs. One might wonder why the routing scheme of Theorem 2 does not apply to or cannot be adapted to directed graphs. The key issue lies in the fact that Lemma 10 holds only for undirected graphs. Without this lemma, the routing process fails in directed graphs. Specifically, in directed

graphs, there exist cases where $u \in C(v)$ and $w \in P(u,v)$, but $w \notin C(v)$. Consequently, even if we store the set $\{\mathtt{L}(u, T(C(v))) \mid v \in B(u)\}$, for every vertex u, we may not be able to route correctly from u to v when $v \in B(u)$.

To overcome this issue, we consider bounded hop diameter graphs, where the hop diameter is the maximum number of edges on a shortest path between any two vertices u, v in the graph, i.e. $D_{hop} = \max_{u,v \in V}(|P(u,v)|)$. In such a case, we can achieve the following theorem.

▶ **Reminder of Theorem 4.** *Let $G = (V, E, w)$ be a weighted directed graph. Let $k \geq 1$ be an integer. There is a $(2k-1)$-stretch compact roundtrip routing scheme that uses local routing tables of size $\tilde{O}(D_{hop} n^{1/k})$, vertex labels of size $\tilde{O}(D_{hop} k)$ and packet headers of size $\tilde{O}(D_{hop})$.*

Proof. The preprocessing algorithm is identical to that of Theorem 2, with one key modification: instead of setting $\mathtt{RT}(u) = \{\mathtt{L}(u, T(C(v))) \mid v \in B(u)\}$, we store $\mathtt{RT}(u) = \{P(u,v) \mid v \in B(u)\}$, where $P(u,v)$ is the entire path from u to v, and similarly instead of setting $\mathtt{L}(u) = \{p_i(u) \mid i \in [k]\}$ we store $\mathtt{L}(u) = \{P(p_i(u), u) \mid i \in [k]\}$.

In the routing algorithm at the source vertex u to a destination vertex v, after determining $\ell = \min\{i \mid p_i(v) \in B(u)\}$. The entire path $P(u, p_\ell(v)) \cup P(p_\ell(v), v)$ is attached to the header to ensure that intermediate vertices can route correctly. The remainder of the proof follows the same steps as in Section 4. ◀

References

1. Ittai Abraham, Cyril Gavoille, Dahlia Malkhi, Noam Nisan, and Mikkel Thorup. Compact name-independent routing with minimum stretch. In Phillip B. Gibbons and Micah Adler, editors, *SPAA 2004: Proceedings of the Sixteenth Annual ACM Symposium on Parallelism in Algorithms and Architectures, June 27-30, 2004, Barcelona, Spain*, pages 20–24. ACM, 2004. doi:10.1145/1007912.1007916.
2. Ittai Abraham, Cyril Gavoille, Dahlia Malkhi, Noam Nisan, and Mikkel Thorup. Compact name-independent routing with minimum stretch. *ACM Trans. Algorithms*, 4(3):37:1–37:12, 2008. doi:10.1145/1367064.1367077.
3. Baruch Awerbuch, Amotz Bar-Noy, Nathan Linial, and David Peleg. Improved routing strategies with succinct tables. *J. Algorithms*, 11(3):307–341, 1990. doi:10.1016/0196-6774(90)90017-9.
4. Baruch Awerbuch and David Peleg. Routing with polynomial communication-space trade-off. *SIAM J. Discret. Math.*, 5(2):151–162, 1992. doi:10.1137/0405013.
5. Ruoxu Cen and Ran Duan. Roundtrip spanners with $(2k-1)$ stretch. *CoRR*, abs/1911.12411, 2019. arXiv:1911.12411.
6. Ruoxu Cen, Ran Duan, and Yong Gu. Roundtrip spanners with (2k-1) stretch. In Artur Czumaj, Anuj Dawar, and Emanuela Merelli, editors, *47th International Colloquium on Automata, Languages, and Programming, ICALP 2020, July 8-11, 2020, Saarbrücken, Germany (Virtual Conference)*, volume 168 of *LIPIcs*, pages 24:1–24:11. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPICS.ICALP.2020.24.
7. Shiri Chechik. Compact routing schemes with improved stretch. In Panagiota Fatourou and Gadi Taubenfeld, editors, *ACM Symposium on Principles of Distributed Computing, PODC '13, Montreal, QC, Canada, July 22-24, 2013*, pages 33–41. ACM, 2013. doi:10.1145/2484239.2484268.
8. Lenore Cowen. Compact routing with minimum stretch. *J. Algorithms*, 38(1):170–183, 2001. doi:10.1006/JAGM.2000.1134.

9 Lenore Cowen and Christopher G. Wagner. Compact roundtrip routing for digraphs. In Robert Endre Tarjan and Tandy J. Warnow, editors, *Proceedings of the Tenth Annual ACM-SIAM Symposium on Discrete Algorithms, 17-19 January 1999, Baltimore, Maryland, USA*, pages 885–886. ACM/SIAM, 1999. URL: http://dl.acm.org/citation.cfm?id=314500.315068.

10 Lenore Cowen and Christopher G. Wagner. Compact roundtrip routing in directed networks. *J. Algorithms*, 50(1):79–95, 2004. doi:10.1016/J.JALGOR.2003.08.001.

11 Tamar Eilam, Cyril Gavoille, and David Peleg. Compact routing schemes with low stretch factor. *J. Algorithms*, 46(2):97–114, 2003. doi:10.1016/S0196-6774(03)00002-6.

12 Pierre Fraigniaud and Cyril Gavoille. A space lower bound for routing in trees. In Helmut Alt and Afonso Ferreira, editors, *STACS 2002, 19th Annual Symposium on Theoretical Aspects of Computer Science, Antibes - Juan les Pins, France, March 14-16, 2002, Proceedings*, volume 2285 of *Lecture Notes in Computer Science*, pages 65–75. Springer, 2002. doi:10.1007/3-540-45841-7_4.

13 Cyril Gavoille, Christian Glacet, Nicolas Hanusse, and David Ilcinkas. On the communication complexity of distributed name-independent routing schemes. In Yehuda Afek, editor, *Distributed Computing - 27th International Symposium, DISC 2013, Jerusalem, Israel, October 14-18, 2013. Proceedings*, volume 8205 of *Lecture Notes in Computer Science*, pages 418–432. Springer, 2013. doi:10.1007/978-3-642-41527-2_29.

14 Alina Harbuzova, Ce Jin, Virginia Vassilevska Williams, and Zixuan Xu. Improved roundtrip spanners, emulators, and directed girth approximation. In David P. Woodruff, editor, *Proceedings of the 2024 ACM-SIAM Symposium on Discrete Algorithms, SODA 2024, Alexandria, VA, USA, January 7-10, 2024*, pages 4641–4669. SIAM, 2024. doi:10.1137/1.9781611977912.166.

15 Avi Kadria and Liam Roditty. Compact routing schemes in undirected and directed graphs. *arXiv preprint arXiv:2503.13753*, 2025. doi:10.48550/arXiv.2503.13753.

16 Goran Konjevod, Andréa W. Richa, and Donglin Xia. Optimal-stretch name-independent compact routing in doubling metrics. In Eric Ruppert and Dahlia Malkhi, editors, *Proceedings of the Twenty-Fifth Annual ACM Symposium on Principles of Distributed Computing, PODC 2006, Denver, CO, USA, July 23-26, 2006*, pages 198–207. ACM, 2006. doi:10.1145/1146381.1146412.

17 Kofi A. Laing. Name-independent compact routing in trees. *Inf. Process. Lett.*, 103(2):57–60, 2007. doi:10.1016/J.IPL.2007.02.015.

18 Jakub Pachocki, Liam Roditty, Aaron Sidford, Roei Tov, and Virginia Vassilevska Williams. Approximating cycles in directed graphs: Fast algorithms for girth and roundtrip spanners. In Artur Czumaj, editor, *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018, New Orleans, LA, USA, January 7-10, 2018*, pages 1374–1392. SIAM, 2018. doi:10.1137/1.9781611975031.91.

19 David Peleg and Eli Upfal. A trade-off between space and efficiency for routing tables. *J. ACM*, 36(3):510–530, 1989. doi:10.1145/65950.65953.

20 Liam Roditty, Mikkel Thorup, and Uri Zwick. Deterministic constructions of approximate distance oracles and spanners. In Luís Caires, Giuseppe F. Italiano, Luís Monteiro, Catuscia Palamidessi, and Moti Yung, editors, *Automata, Languages and Programming, 32nd International Colloquium, ICALP 2005, Lisbon, Portugal, July 11-15, 2005, Proceedings*, volume 3580 of *Lecture Notes in Computer Science*, pages 261–272. Springer, 2005. doi:10.1007/11523468_22.

21 Liam Roditty, Mikkel Thorup, and Uri Zwick. Roundtrip spanners and roundtrip routing in directed graphs. *ACM Trans. Algorithms*, 4(3):29:1–29:17, 2008. doi:10.1145/1367064.1367069.

22 Liam Roditty and Roei Tov. New routing techniques and their applications. In Chryssis Georgiou and Paul G. Spirakis, editors, *Proceedings of the 2015 ACM Symposium on Principles of Distributed Computing, PODC 2015, Donostia-San Sebastián, Spain, July 21 - 23, 2015*, pages 23–32. ACM, 2015. doi:10.1145/2767386.2767409.

23 Eli Stafford and Chunjiang Zhu. Improved sourcewise roundtrip spanners with constant stretch. In Weili Wu and Guangmo Tong, editors, *Computing and Combinatorics - 29th International Conference, COCOON 2023, Hawaii, HI, USA, December 15-17, 2023, Proceedings, Part I*, volume 14422 of *Lecture Notes in Computer Science*, pages 297–309. Springer, 2023. doi:10.1007/978-3-031-49190-0_21.

24 Mikkel Thorup and Uri Zwick. Compact routing schemes. In Arnold L. Rosenberg, editor, *Proceedings of the Thirteenth Annual ACM Symposium on Parallel Algorithms and Architectures, SPAA 2001, Heraklion, Crete Island, Greece, July 4-6, 2001*, pages 1–10. ACM, 2001. doi:10.1145/378580.378581.

25 Mikkel Thorup and Uri Zwick. Approximate distance oracles. *J. ACM*, 52(1):1–24, 2005. doi:10.1145/1044731.1044732.

A 7-stretch directed roundtrip routing with $|\mathtt{RT}(u)| = O(n^{1/3})$

In this section, we consider roundtrip routing in general weighted directed graphs. Roditty et. al [21] obtained a roundtrip routing scheme with $(4k+\varepsilon)$-stretch and routing tables of size $\tilde{O}(\frac{1}{\varepsilon}n^{1/k}\log W)$. If we set $k=3$ we get a $(12+\varepsilon)$-stretch roundtrip routing scheme with routing tables of size $\tilde{O}(\frac{1}{\varepsilon}n^{1/3}\log W)$. In this section, we present a 7-stretch roundtrip routing scheme with routing tables of size $\tilde{O}(n^{1/3})$. We prove:

▶ **Reminder of Theorem 3.** *Let $G=(V,E,w)$ be a weighted directed graph. There is a 7-stretch compact roundtrip routing scheme that uses local routing tables of size $\tilde{O}(n^{1/3})$, vertex labels of size $\tilde{O}(1)$ and packet headers of size $\tilde{O}(1)$.*

First, we describe the preprocessing algorithm, which computes the routing tables and assigns vertex labels. The input to the algorithm is a graph $G=(V,E,w)$. The algorithm uses Lemma 6 and Lemma 5 to build a hierarchy of vertex sets $V=A_0\supseteq A_1\supseteq A_2\supseteq A_3=\emptyset$, where $|A_i|=n^{1-i/k}$. For every $u\in V$ it holds that $|C(u,A_1)|=\tilde{O}(n^{1/3})$ and $|B_i(u)|=\tilde{O}(n^{1/k})$, where $0\le i\le 3$. Next, for every $u\in V$, the preprocessing algorithm computes $B(u)$ and $C(u)$. Then, for every $u\in V$, the algorithm sets the routing table $\mathtt{RT}(u)$ to:

$$\mathtt{RT}(u)=\begin{cases} L(w,T(B_0(u))) & \text{for every } w\in B_0(u)\\ L(u,T(B_0(w))) & \text{for every } w \text{ s.t. } u\in T(B_0(w))\\ L(u,T(C(w))) & \text{for every } w\in A_2\\ L(u,T(C(w))) & \text{for every } w\in B_1(u), \text{ if } P(u,w)\subseteq C(w)\\ L(w,T(C(p_2(z)))) & \text{for every } w\in B_1(u), z=\arg\min_{x\in P(u,w),x\notin C(w)}\{d(u,x)\}\end{cases}$$

and the label $\mathtt{L}(u)$ to:

$$\mathtt{L}(u)=\{\mathtt{L}(u,T(B_0(u))),\mathtt{L}(p_1(u),T(B_0(u))),\mathtt{L}(u,T(C(p_2(u))))\}$$

We now bound the size of the routing tables and the vertex labels.

▶ **Lemma 17.** *$|\mathtt{RT}(u)|=\tilde{O}(n^{1/3})$ and $|\mathtt{L}(u)|=\tilde{O}(1)$*

Proof. From Lemma 5 it follows that $|B_i(u)|=\tilde{O}(n^{1/3})$, for every $1\le i\le 2$. In addition we get that $|A_2|=\tilde{O}(n^{1/3})$. From Lemma 6 it follows that $|B_0(u)|=\tilde{O}(n^{1/3})$ and $|C(u,A_1)|=\tilde{O}(n^{1/3})$. From Lemma 11 it follows that for every tree T it holds that $|\mathtt{L}(u,T)|=\tilde{O}(1)$. Therefore, $|\mathtt{RT}(u)|\le\tilde{O}(1)\cdot(|B_0(u)|+|C(u,A_1)|+|A_2|+|B_1(u)|+|B_1(u)|)=\tilde{O}(n^{1/3})$, as required.

The label of each vertex is composed of three tree labels. From Lemma 11 it follows that each tree label is of size $\tilde{O}(1)$. Therefore, $|\mathtt{L}(u)|=3\cdot\tilde{O}(1)=\tilde{O}(1)$, as required. ◀

Algorithm 1 `Route`(u, v).

```
1  if v ∈ B_0(u) then Route from u to v on T(B_0(u)) ;
2  if v ∈ C(u, A_1) then Route from u to v on T(B_0(v)) ;
3  if p_1(v) ∈ B_1(u) then
4      if P(u, p_1(v)) ⊆ C(p_1(v)) then
5          Route from u to p_1(v) on T(C(p_1(v))) and then route from p_1(v) to v on
             T(B_0(v)).
6      else
7          z ← arg min_{x∈P(u,p_1(v)),x∉C(p_1(v))}{d(u, x)};
8          Route from u to p_1(v) on T(C(p_2(z))) and then route from p_1(v) to v on
             T(B_0(v)).
9  else
10     Route from u to v on T(C(p_2(v))).
```

The routing algorithm routes a message from u to v as follows. If $v \in B_0(u)$, then the algorithm routes the message using the tree $T(B_0(u))$. Otherwise, if $v \in C(u, A_1)$ then the algorithm routes the message using the tree $T(B_0(v))$. If $v \notin B_0(u)$ and $v \notin C(u, A_1)$, then the algorithm checks if $p_1(v) \in B_1(u)$. In this case, if $P(u, p_1(v)) \subseteq C(p_1(v))$ then the algorithm routes on $T(C(p_1(v)))$ from u to $p_1(v)$, and then from $p_1(v)$ to v on the tree $T(B_0(v))$ (see Figure 2 (a)). Otherwise, if $P(u, p_1(v)) \not\subseteq C(p_1(v))$, then the algorithm routes from u to $p_1(v)$ on the tree $T(C(p_2(z)))$, where $z = \arg\min_{x \in P(u,p_1(v)), x \notin C(p_1(v))}\{d(u, x)\}$, and then the algorithm routes from $p_1(v)$ to v on the tree $T(B_0(v))$ (see Figure 2 (b)).

Finally, if $p_1(v) \notin B_1(u)$, then the algorithm routes from u to v on the tree $T(C(p_2(v)))$ (see Figure 2 (c)). A pseudo-code for the routing algorithm is given in Algorithm 1.

In the next lemma, we show that all intermediate vertices have the necessary information to complete the routing process once the routing tree is determined.

▶ **Lemma 18.** *The following properties hold:*

1. *If $v \in B_0(u)$, then for every $w \in P(u, v)$, we have $\mathtt{L}(w, T(B_0(u))) \in \mathtt{RT}(w)$.*
2. *If $v \in C(u, A_1)$, then for every $z \in P(u, v)$, we have $\mathtt{L}(z, T(B_0(v))) \in \mathtt{RT}(z)$.*
3. *If $p_1(v) \in B_1(u)$ and $P(u, p_1(v)) \subseteq C(v)$, then for every $z \in P(u, p_1(v))$, we have $\mathtt{L}(z, T(C(p_1(v)))) \in \mathtt{RT}(z)$.*
4. *If $w \in A_2$, then for every $z \in P(u, w) \cup P(w, v)$, we have $\mathtt{L}(z, T(C(w))) \in \mathtt{RT}(z)$.*
5. *For every $w \in P(p_1(v), v)$, we have $\mathtt{L}(w, B_0(v)) \in \mathtt{RT}(w) \cup \mathtt{L}(v)$.*

Proof. From Lemma 7, we know that for every $v \in B(u, X)$ and $w \in P(u, v)$, we have that $w \in B(u, X)$. Therefore, properties 1 and 2 hold.

For property 3, we have that the entire path $P(u, p_1(v))$ is contained in $C(p_1(v))$. Hence, any sub-path of $P(u, p_1(v))$ is also in $C(p_1(v))$, and we have $\mathtt{L}(z, T(C(p_1(v)))) \in \mathtt{RT}(z)$ for every $z \in P(u, p_1(v))$.

For property 4, since $w \in A_2$, we know that $C(w) = V$. Therefore, all vertices in the graph (and specifically z) hold $\mathtt{L}(z, T(C(w))$.

For property 5, for $p_1(v)$, we have $\mathtt{L}(p_1(v), T(B_0(v))) \in \mathtt{L}(v)$. For every other $w \in P(p_1(v), v)$, we have from Lemma 13 that $v \in C(w)$ and therefore $w \in T(B(v))$, and we have $\mathtt{L}(w, T(B_0(v)))$ in $\mathtt{RT}(w)$, as required. ◀

We now turn to prove that the stretch of the compact roundtrip routing scheme is 7.

▶ **Lemma 19.** $\hat{d}(u \leftrightarrow v) \leq 7 \cdot d(u \leftrightarrow v)$

Proof. First, consider the case that $v \in B_0(u)$. In this case, the routing algorithm routes from u to v along $P(u, v)$, so $\hat{d}(u, v) = d(u, v)$. Since $v \in B_0(u)$, by the definition of $C(v, A_1)$, we know that $u \in C(v, A_1)$. Therefore, the routing algorithm from v to u follows $P(v, u$. Thus,

$$\hat{d}(u \leftrightarrow v) = \hat{d}(u, v) + \hat{d}(v, u) = d(u, v) + d(v, u) = d(u \leftrightarrow v),$$

as required. Similarly, if $v \in C(u, A_1)$, using symmetrical arguments, we get that $\hat{d}(u \leftrightarrow v) = d(u \leftrightarrow v)$, as required.

Next, consider the case that $v \notin B_0(u)$ and $v \notin C(u, A_1)$. In the following claim, we show that in this case, $\hat{d}(u, v) \leq d(u, v) + 3d(u \leftrightarrow v)$. Using this claim, we obtain:

$$\hat{d}(u \leftrightarrow v) = \hat{d}(u, v) + \hat{d}(v, u) \leq d(u, v) + 3d(u \leftrightarrow v) + d(v, u) + 3d(u \leftrightarrow v) \leq 7d(u \leftrightarrow v),$$

as required.

▷ Claim 19.1. If $v \notin B_0(u)$ and $v \notin C(u, A_1)$, then:

$$\hat{d}(u, v) \leq d(u, v) + 3d(u \leftrightarrow v).$$

Proof. Since $v \notin B_0(u)$ and $v \notin C(u, A_1)$, we know from the definitions of $B_0(u)$ and $C(u, A_1)$ that:

$$d(u \leftrightarrow p_1(u)) \leq d(u \leftrightarrow v) \quad \text{and} \quad d(v \leftrightarrow p_1(v)) \leq d(u \leftrightarrow v). \tag{1}$$

We divide the proof into two cases: the case that $p_1(v) \notin B(u)$ and the case that $p_1(v) \in B(u)$. If $p_1(v) \notin B(u)$ we can apply Lemma 9 and get:

$$d(v \leftrightarrow p_2(v)) \leq 2d(u \leftrightarrow v) + d(v \leftrightarrow p_1(v)) \overset{(1)}{\leq} 3d(u \leftrightarrow v). \tag{2}$$

Since $p_1(v) \notin B(u)$, the routing algorithm routes from u to v on the tree $T(C(p_2(v))$. Therefore, we have:

$$\begin{aligned}\hat{d}(u, v) &= d_{T(C(p_2(v)))}(u, v) \leq d(u, p_2(v)) + d(p_2(v), v) \\ &\overset{\triangle}{\leq} d(u, v) + d(v \leftrightarrow p_2(v)) \overset{(2)}{\leq} d(u, v) + 3d(u \leftrightarrow v),\end{aligned}$$

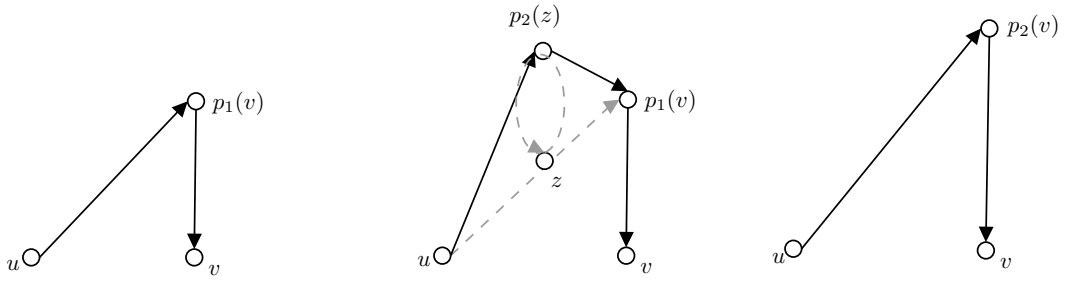
as required.

Next, consider the case that $p_1(v) \in B_1(u)$. If $P(u, p_1(v)) \subseteq C(p_1(v))$, then the routing algorithm routes along the shortest path from u to $p_1(v)$. Therefore, we have:

$$\hat{d}(u, v) \leq d(u, p_1(v)) + d(p_1(v), v) \overset{\triangle}{\leq} d(u, v) + d(v \leftrightarrow p_1(v)) \overset{(1)}{\leq} d(u, v) + d(u \leftrightarrow v),$$

as required.

Otherwise, if $P(u, p_1(v)) \not\subseteq C(p_1(v))$, let z be the first vertex in $P(u, p_1(v))$ such that $z \notin C(p_1(v))$. From the definition of $C(p_1(v))$, it follows that $p_1(v) \notin B_1(z$. From the definition of $B_1(z)$ we get that $d(z \leftrightarrow p_2(z)) \leq d(z \leftrightarrow p_1(v))$. Since $z \in P(u, p_1(v))$, it follows that $d(z \leftrightarrow p_1(v)) \leq d(u \leftrightarrow p_1(v))$. Combining these inequalities, we get that: $d(z \leftrightarrow p_2(z)) \leq d(u \leftrightarrow p_1(v))$.



(a) $p_1(v) \in B_1(u)$
$P(u, p_1(v)) \subseteq C(p_1(v))$

(b) $p_1(v) \in B_1(u)$
$z \notin C(p_1(v))$
$z \in P(u, p_1(v))$

(c) $p_1(v) \notin B_1(u)$

Figure 2 The routing of Theorem 3. $v \notin B_0(u) \cup C(u, A_1)$.

In this case, the routing algorithm routes from u to $p_1(v)$ on the tree $T(C(p_2(z))$, and then from $p_1(v)$ to v. We get that:

$$\begin{aligned}
\hat{d}(u,v) &= d_{T(C(p_2(z)))}(u, p_1(v)) + d(p_1(v), v) \\
&\leq d(u, p_2(z)) + d(p_2(z), p_1(v)) + d(p_1(v), v) \\
&\overset{\triangle}{\leq} d(u,z) + d(z, p_2(z)) + d(p_2(z), z) + d(z, p_1(v)) + d(p_1(v), v) \\
&= d(u,z) + d(z \leftrightarrow p_2(z)) + d(z, p_1(v)) + d(p_1(v), v).
\end{aligned}$$

Since $z \in P(u, p_1(v))$, we know that $d(u,z) + d(z, p_1(v)) = d(u, p_1(v))$. Recall that $d(z \leftrightarrow p_2(z)) \leq d(u \leftrightarrow p_1(v))$. Therefore:

$$\begin{aligned}
\hat{d}(u,v) &\leq d(u,z) + d(z, p_1(v)) + d(p_1(v), v) + d(z \leftrightarrow p_2(z)) \\
&\leq d(u, p_1(v)) + d(p_1(v), v) + d(u \leftrightarrow p_1(v)) \\
&\overset{\triangle}{\leq} d(u,v) + d(v \leftrightarrow p_1(v)) + d(u \leftrightarrow v) + d(v \leftrightarrow p_1(v)) \\
&\overset{(1)}{\leq} d(u,v) + d(u \leftrightarrow v) + d(u \leftrightarrow v) + d(u \leftrightarrow v) = d(u,v) + 3d(u \leftrightarrow v),
\end{aligned}$$

as required. ◁

From Claim 19.1, we have that $\hat{d}(u,v) \leq d(u,v) + 3d(u \leftrightarrow v)$ and $\hat{d}(v,u) \leq d(u,v) + 3d(u \leftrightarrow v)$. Therefore:

$$\hat{d}(u \leftrightarrow v) = \hat{d}(u,v) + \hat{d}(v,u) \leq d(u,v) + 3d(u \leftrightarrow v) + d(v,u) + 3d(u \leftrightarrow v) = 7d(u \leftrightarrow v),$$

as required. ◀

Theorem 3 follows from Lemma 17, Lemma 18 and Lemma 19.