

# Compact Routing in Unit Disk Graphs

Wolfgang Mulzer 

Institut für Informatik, Freie Universität Berlin, 14195 Berlin, Germany  
mulzer@inf.fu-berlin.de

Max Willert

Institut für Informatik, Freie Universität Berlin, 14195 Berlin, Germany  
willerma@inf.fu-berlin.de

---

## Abstract

Let  $V \subset \mathbb{R}^2$  be a set of  $n$  sites in the plane. The *unit disk graph*  $DG(V)$  of  $V$  is the graph with vertex set  $V$  where two sites  $v$  and  $w$  are adjacent if and only if their Euclidean distance is at most 1.

We develop a *compact routing scheme*  $\mathcal{R}$  for  $DG(V)$ . The routing scheme  $\mathcal{R}$  preprocesses  $DG(V)$  by assigning a *label*  $\ell(v)$  to every site  $v$  in  $V$ . After that, for any two sites  $s$  and  $t$ , the scheme  $\mathcal{R}$  must be able to route a packet from  $s$  to  $t$  as follows: given the label of a *current vertex*  $r$  (initially,  $r = s$ ), the label of the target vertex  $t$ , and additional information in the header of the packet, the scheme determines a neighbor  $r'$  of  $r$ . Then, the packet is forwarded to  $r'$ , and the process continues until the packet reaches its desired target  $t$ . The resulting path between the source  $s$  and the target  $t$  is called the *routing path* of  $s$  and  $t$ . The *stretch* of the routing scheme is the maximum ratio of the total Euclidean length of the routing path and of the shortest path in  $DG(V)$ , between any two sites  $s, t \in V$ .

We show that for any given  $\varepsilon > 0$ , we can construct a routing scheme for  $DG(V)$  with diameter  $D$  that achieves stretch  $1 + \varepsilon$ , has label size  $(1/\varepsilon)^{O(\varepsilon^{-2})} \log D \log^3 n / \log \log n$ , and the header has at most  $O(\log^2 n / \log \log n)$  bits. In the past, several routing schemes for unit disk graphs have been proposed. Our scheme achieves poly-logarithmic label and header size, small stretch and does not use any neighborhood oracles.

**2012 ACM Subject Classification** Theory of computation  $\rightarrow$  Computational geometry

**Keywords and phrases** routing scheme, unit disk graph, separator

**Digital Object Identifier** 10.4230/LIPIcs.ISAAC.2020.40

**Related Version** <https://arxiv.org/abs/2002.10841>

**Funding** *Wolfgang Mulzer*: Partially supported by ERC STG 757609.

## 1 Introduction

The *routing problem* is a well-known problem in distributed graph algorithms [13, 17]. We are given a graph  $G$  and want to preprocess it by assigning *labels* to each node of  $G$  such that the following task can be solved: a data packet is located at a source node and has to be routed to a target node. A routing scheme should have several properties. First, routing must be *local*: a node uses only the header of the packet (including the label of the target node) and its own local information to compute a neighbor to which the packet is sent next. Second, the routing should be *efficient*: the ratio of the routed path and the shortest path — the *stretch factor* — should be close to 1. Finally, the scheme should be *compact*: the size of the labels (in bits) and headers must be small. In the literature, one can find many different techniques and models for routing. A common tool is called the *routing table*. A routing table is a sequence of bits stored in a node. Typically, routing tables contain more information about the topology of the graph and are different from labels. In this article, we do not use routing tables, but store all the information in the labels. Moreover, the header moves with the data packet through the graph. It can be split into two different parts: the



© Wolfgang Mulzer, Max Willert;  
licensed under Creative Commons License CC-BY

31st International Symposium on Algorithms and Computation (ISAAC 2020).

Editors: Yixin Cao, Siu-Wing Cheng, and Minming Li; Article No. 40; pp. 40:1–40:14

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

45 first part is called *static header* and contains the target label. The static part of the header  
 46 is immutable. On the other hand, the *dynamic header* contains mutable information. The  
 47 existence of dynamic headers make it possible to implement recursive routing strategies or to  
 48 remember information from past positions of the packet.

49 Furthermore, the literature distinguishes two types of input models. In the *fixed-port*  
 50 *model*, the given graph already has a complete list of *ports* for each node  $v$ , i.e., a fixed  
 51 numbering of the neighbors of  $v$  used to identify the next hop of the packet. In particular,  
 52 it is not possible to renumber the ports. In contrast, the *designer-port model* allows us to  
 53 assign arbitrary port numbers during the preprocessing, see [11,12,22]. Below, we will briefly  
 54 discuss the advantages and disadvantages of these two models.

55 A trivial solution to solve the routing problem is to store the complete shortest path tree  
 56 in every label. Then it is easy to route the data packets along a shortest path. However,  
 57 such a routing scheme is not compact. Moreover, Peleg and Upfal [17] proved that in general  
 58 graphs, any routing scheme that achieves a constant stretch factor must store at least a  
 59 polynomial number of bits in some nodes.

60 Nevertheless, there is a rich collection of routing schemes for general graphs [1,3,7,9,10,  
 61 18,19]. For example, the scheme by Roditty and Tov [19] uses labels of size  $mn^{O(1/\sqrt{\log n})}$   
 62 and routes a packet from  $s$  to  $t$  on a path of length  $O(k\Delta + m^{1/k})$ , where  $\Delta$  is the shortest  
 63 path distance between  $s$  and  $t$ ,  $k > 2$  is any fixed integer,  $n$  is the number of nodes, and  $m$  is  
 64 the number of edges. Their routing scheme needs headers of poly-logarithmic size. The lower  
 65 bound result by Peleg and Upfal [17] shows that it is hopeless to find efficient routing schemes  
 66 for general graphs that are compact as well, meaning that at most a poly-logarithmic number  
 67 of bits in the labels/tables are necessary. Thus, it is natural to investigate special interesting  
 68 graph classes and to develop compact and efficient routing schemes for them. For example, it  
 69 is possible to route in trees along a shortest path by using a poly-logarithmic number of bits  
 70 in the label [11,20,22]. Moreover, in planar graphs, for any fixed  $\varepsilon > 0$ , we can find a routing  
 71 scheme that achieves the stretch factor  $1 + \varepsilon$ . Again, the number of bits for the labels is  
 72 poly-logarithmic [21]. The same holds for visibility graphs of simple polygons [4]. Moreover,  
 73 see [2,15] for different compact routing schemes in networks with low doubling dimension.

74 Our graph class of interest comes from the study of mobile and wireless networks. These  
 75 networks are usually modeled as *unit disk graphs* [8]. Nodes in this network are points in the  
 76 plane and two nodes are connected if their distance is at most one. This is equivalent to  
 77 a disk intersection graph in which all disks have diameter one. For unit disk graphs there  
 78 are known routing schemes. The first routing scheme is by Kaplan et al. [14] and uses the  
 79 fixed-port model. They present a routing scheme with stretch  $1 + \varepsilon$  and routing table size  
 80  $O(\log^2 n \log^2 D)$ , where  $D$  is the diameter of the given unit disk graph. Their routing is  
 81 recursive and needs an additional header of size  $O(\log n \log D)$ . The second routing scheme is  
 82 due to Yan, Xiang, and Dragan [24]. They present a routing scheme with label size  $O(\log^2 n)$   
 83 and show that a data packet routes along a path of length at most  $5\Delta + 13$ , where  $\Delta$  is the  
 84 length of the optimal path. The designer-port model is used.

85 Here, we present a compact routing scheme that achieves stretch  $1 + \varepsilon$ . We obtain label  
 86 size  $(1/\varepsilon)^{O(\varepsilon^{-2})} \log D \log^3 n / \log \log n$ . We use the fixed-port model. Moreover, we do not  
 87 use neighborhood oracles and the dynamic part of the header is empty except for the case  
 88 that the current and target vertices are very close. Here, the dynamic header size is at most  
 89  $O(\log^2 n / \log \log n)$ . In the conclusion, we will discuss how our scheme compares to the other  
 90 schemes.

## 2 Preliminaries

We explain our graph-theoretic notation and discuss how the routing scheme can access the input graph. Then, we provide a precise definition of our notion of a routing scheme and give some background on unit disk graphs.

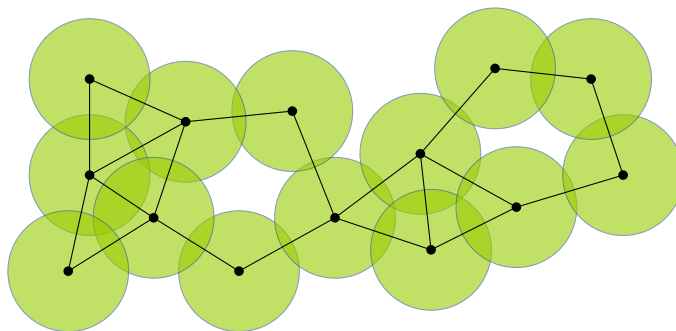
We are given a *simple* and *undirected* graph  $G = (V, E)$  with  $n$  vertices. The edges are weighted by a non-negative weight function  $w: E \rightarrow \mathbb{R}_0^+$ . We write  $d_G(s, t)$  for the (weighted) *shortest path distance* between the vertices  $s, t \in V$  and we omit the subscript  $G$  if it follows from the context. Throughout the whole article we assume that the graph is *connected*.

**Graph Access Model.** Let  $\Sigma = \{0, 1\}$ , and  $[m] = \{0, 1, \dots, m\}$ , for  $m \in \mathbb{N}$ . We explain how the routing scheme may access the input graph  $G = (V, E)$ . Every vertex  $v \in V$  has an identifier  $v_{\text{id}} \in \Sigma^+$  of length  $|v_{\text{id}}| = \lceil \log n \rceil$ . We use the *fixed-port model* [11, 12, 22]. In this model the port numbers are assigned arbitrarily. The neighbors of a vertex  $v \in V$  are accessed through *ports*. More precisely, there is a partial function  $\text{node}: V \times [n-1] \rightarrow V$ , that assigns to every vertex  $v \in V$  and to every port number  $p \in [n-1]$  the neighbor  $w = \text{node}(v, p)$  that can be reached through the port  $p$  at vertex  $v$ .

Other authors also use the *designer-port model* [11, 22, 24]. In this model, the routing scheme can determine the assignment of port numbers to the incident edges of each vertex  $v \in V$  during the preprocessing phase. This additional power in the model can lead to more efficient routing schemes [11, 12, 22, 24]. However, a routing scheme that uses the designer-port cannot easily be used as a building block for more complicated routing schemes, since additional lookup tables become necessary in order to store the assignments of the port numbers.

**Routing Schemes.** Let  $\mathcal{G}$  be a graph class. A *routing scheme*  $\mathcal{R}$  for  $\mathcal{G}$  consists of a family of *labeling functions*  $\ell_G: V(G) \rightarrow \Sigma^+$ , for each  $G \in \mathcal{G}$ . The labeling function  $\ell_G$  assigns a bit string  $\ell_G(v)$  to every node  $v$  of  $G \in \mathcal{G}$ . The label  $\ell_G(v)$  serves as the address of the node  $v \in V$  in  $G$ . In contrast to the identifier of a node, the label usually contains the identifier, but some more information about the topology of the graph  $G$ . While the identifier is given as fixed input, the label is chosen by the routing scheme during the preprocessing. As before, we omit the index  $G$  if the context is clear. Furthermore,  $\mathcal{R}$  has a *routing function*  $\sigma: \Sigma^+ \times \Sigma^+ \times \Sigma^+ \rightarrow \mathbb{N} \times \Sigma^+$ . The routing function  $\sigma$  describes the behavior of the routing scheme, as follows: assume a data packet with dynamic header  $h$  is located at a vertex  $s \in V$  and must be routed to a destination  $t \in V$ . Then,  $\sigma(\ell(s), \ell(t), h)$  computes a port  $p \in \mathbb{N}$  and a new header  $h'$  so that the next hop of the data packet (attached with the new header  $h'$ ) is from  $s$  to  $\text{node}(s, p)$ . Now, let  $v_0 = s$ ,  $h_0 = \varepsilon$ ,  $(p_i, h_{i+1}) = \sigma(\ell(v_i), \ell(t), h_i)$ , and  $v_{i+1} = \text{node}(v_i, p_i)$  for  $i \geq 0$ . The sequence  $(v_i)_{i \in \mathbb{N}}$  is called the *routing sequence*. The routing scheme  $\mathcal{R}$  is *correct*, for  $G \in \mathcal{G}$ , if and only if for all distinct  $s, t \in V(G)$ , there is a number  $m(s, t) \in \mathbb{N}$  such that  $v_j = t$ , for all  $j \geq m(s, t)$ , and  $v_j \neq t$ , for all  $j = 0, \dots, m(s, t) - 1$ . If  $\mathcal{R}$  is correct for  $G = (V, E)$ , then  $\delta_G(s, t) = \sum_{i=1}^{m(s,t)} w(v_{i-1}, v_i)$  is called the *routing length* between  $s$  and  $t$  (in  $G$ ). The *stretch* of the routing scheme is the largest ratio  $\delta_G(s, t)/d_G(s, t)$  over all distinct vertices  $s, t \in V$ . The goal is to achieve a routing scheme that minimizes the stretch factor as well as the number of bits stored in the labels and the headers.

**Unit Disk Graphs.** Our graph class of interest are the *unit disk graphs*. Let  $V \subset \mathbb{R}^2$  be a set of  $n$  points in the Euclidean plane. The unit disk graph  $\text{DG}(V)$  of  $V$  has vertex set  $V$  and an edge between two vertices  $v, w \in V$  if and only if the *Euclidean distance*  $|vw|$  is at



■ **Figure 1** The disks in the unit disk graph have diameter 1 and there is an edge between two midpoints if and only if their corresponding disks intersect.

135 most 1, see Figure 1. The weight of the edge  $vw$  is  $|vw|$ . Throughout, we will assume that  
 136  $DG(V)$  is connected, and we will use  $D$  to denote the diameter  $\max_{v,w \in V} d(v, w)$  of  $DG(V)$ .  
 137 Clearly, we have  $D \leq n - 1$ .

### 138 3 Building Blocks

139 The overall idea for our routing scheme is as follows: We use a hierarchy of sparse covers to  
 140 cover the unit disk graph with  $O(\log D \log n)$  connected subgraphs. For the subgraphs with  
 141 large diameter we use a recently developed distance oracle of Chan and Skrepetos and turn  
 142 it into an efficient routing scheme with *additive* stretch. Here we use the fact, that we can  
 143 route easily in trees. For the subgraphs with small diameter we use a routing scheme, that is  
 144 used for graphs with low doubling dimension.

145 In this section we review the routing schemes for trees and graphs with low doubling  
 146 dimension and adapt the distance oracle to a new routing scheme. In the following section  
 147 we combine these building blocks to obtain our result.

#### 148 3.1 Routing in Trees and Graphs with Low Doubling Dimension

149 The first routing scheme is for trees. There are many different such schemes, based on similar  
 150 ideas. We would like to point out that some of these routing schemes can achieve label  
 151 size  $O(\log n)$ , see [11, 22]. However, these routing schemes work only in the designer-port  
 152 model and therefore are not useful as building blocks for more complex routing schemes,  
 153 especially if—as in our routing case—we need to be able to route in several subtrees of the  
 154 input graph.<sup>1</sup> The following lemma is due to Fraigniaud and Gavoille [11] as well as Thorup  
 155 and Zwick [22].

156 ► **Lemma 3.1.** *Let  $T$  be an  $n$ -vertex tree with arbitrary edge weights. There is a routing  
 157 scheme for  $T$  with label size  $O(\log^2 n / \log \log n)$  whose routing function  $\sigma_{\text{tree}}$  sends a data  
 158 packet along a shortest path, for any pair of vertices. The dynamic part of the header is  
 159 empty.*

<sup>1</sup> In fact, there is a lower bound that shows that label size  $O(\log n)$  cannot be achieved in the fixed-port model [12].

160 The second routing scheme is efficient for unit disk graphs with small diameter. Let  
 161  $G = (V, E)$  be a simple, undirected, weighted graph. For  $v \in V$  and  $r > 0$ , we define  
 162  $B(v, r) = \{w \in V \mid d(v, w) \leq r\}$  as the *ball of  $v$  with radius  $r$* . The *doubling dimension* of a  
 163 graph is the smallest value  $\alpha$  such that any ball  $B(v, r)$  can be covered by at most  $2^\alpha$  balls  
 164 of radius at most  $r/2$ . The following lemma is due to Konjevod, Richa, and Xia [15].

165 ► **Lemma 3.2.** *Let  $G$  be an  $n$ -vertex graph with doubling dimension  $\alpha$ . Furthermore, let*  
 166  $\varepsilon > 0$ . *There is a routing scheme with label size  $(1/\varepsilon)^{O(\alpha)} \log^3 n$  and dynamic header size*  
 167  $O(\log^2 n / \log \log n)$ , *whose routing function achieves stretch factor  $1 + \varepsilon$ .*

168 The following lemma bounds the doubling dimension of a unit disk graph in terms of  $D$ .

169 ► **Lemma 3.3.** *Let  $\text{DG}(V)$  be a unit disk graph,  $v \in V$  a vertex, and  $r > 0$ . We can cover*  
 170 *the ball  $B = B(v, r)$  with  $O(\max(1, r^2))$  balls of diameter at most  $r/2$ .*

171 **Proof.** Let  $E \in \mathbb{R}^2$  be the Euclidean disk of radius  $r$  centered at  $v$ . Obviously,  $B \subset E$ .  
 172 Moreover, the Euclidean disk  $E$  can be covered by a set  $\mathcal{E}$  of  $K = O(\max(1, r^2))$  Euclidean  
 173 disks each of radius  $r' = \min(r/4, 1/2)$ . This follows from a simple covering argument. For  
 174 each disk  $E_i \in \mathcal{E}$ , we fix a vertex  $v_i$  as follows: if  $E_i \cap B \neq \emptyset$ , then  $v_i$  is an arbitrary vertex  
 175 of  $E_i \cap B$ . Otherwise, if  $E_i \cap B = \emptyset$ , we let  $v_i$  be an arbitrary vertex of  $B$ . Since  $r' \leq 1/2$ ,  
 176 the vertices in  $E_i$  form a clique in  $\text{DG}(V)$ . Hence, we have  $E_i \cap B \subseteq B(v_i, 2r')$ . Next, from  
 177  $r' \leq r/4$  we get  $B(v_i, 2r') \subseteq B(v_i, r/2)$ . Thus,

$$178 \quad B(v, r) \subseteq \bigcup_{i=1}^K (E_i \cap B) \subseteq \bigcup_{i=1}^K B(v_i, 2r_i) \subseteq \bigcup_{i=1}^K B(v_i, r/2).$$

179 This finishes the proof. ◀

180 Finally, the routing scheme for unit disk graphs with small diameter follows from  
 181 Lemma 3.2 and Lemma 3.3.

182 ► **Lemma 3.4.** *Let  $\text{DG}(V)$  be an  $n$ -vertex unit disk graph with diameter  $D$ . Furthermore,*  
 183 *let  $\varepsilon > 0$ . There is a routing scheme with label size  $(1/\varepsilon)^{O(D^2)} \log^3 n$  and dynamic header*  
 184 *size  $O(\log^2 n / \log \log n)$ , whose routing function  $\sigma_{\text{diam}}$  achieves stretch factor  $1 + \varepsilon$ .*

## 185 3.2 The Distance Oracle of Chan and Skrepetos

186 Our routing scheme is based on the recent approximate distance oracle for unit disk graphs by  
 187 Chan and Skrepetos [6]: we are given a set  $V \subset \mathbb{R}^2$  of  $n$  points in the plane and a parameter  
 188  $\varepsilon \geq D^{-1}$ , where  $D$  is the diameter of  $\text{DG}(V)$ . Chan and Skrepetos show how to compute in  
 189  $O((1/\varepsilon)^3 n \log^2 n)$  time a data structure of size  $O((1/\varepsilon)n \log n)$  that can answer *approximate*  
 190 *distance queries* in  $\text{DG}(V)$  in  $O((1/\varepsilon) \log n)$  time: given two vertices  $s, t \in V$ , compute a  
 191 number  $\theta \in \mathbb{R}$  with  $d(s, t) \leq \theta \leq d(s, t) + O(\varepsilon D)$ . The main tool for this data structure is a  
 192 suitable hierarchical decomposition of  $\text{DG}(V)$ . More precisely, Chan and Skrepetos show  
 193 that given  $V$ , one can compute in  $O(n \log n + (1/\varepsilon)n)$  time a *decomposition tree*  $\mathcal{T}$  for  $\text{DG}(V)$   
 194 with the following properties.<sup>2</sup>

<sup>2</sup> The reader familiar with the work of Chan and Skrepetos may notice that we have slightly extended the notion of portals: while Chan and Skrepetos define portals only for inner nodes, we also define portals for the leaves. This does not change the essence of the decomposition, but makes the presentation more unified.

## 40:6 Compact Routing in Unit Disk Graphs

- 195 ■ Every node  $\mu$  of  $\mathcal{T}$  is assigned two sets  $V(\mu)$  and  $\text{port}(\mu)$  satisfying  $\text{port}(\mu) \subseteq V(\mu) \subseteq V$ .
- 196 The subgraph of  $\text{DG}(V)$  induced by  $V(\mu)$  is connected and the vertices in  $\text{port}(\mu)$  are
- 197 called *portals*.
- 198 ■ If  $\mu$  is the root, then  $V(\mu) = V$ .
- 199 ■ If  $\mu$  is an inner node with  $k$  children  $\sigma_1, \dots, \sigma_k$ , the sets  $\text{port}(\mu), V(\sigma_1), \dots, V(\sigma_k)$  are
- 200 pairwise disjoint, and we have  $V(\sigma_i) \subseteq V(\mu)$ , for  $1 \leq i \leq k$ .
- 201 ■ If  $\mu$  is a leaf, then  $V(\mu) = \text{port}(\mu)$ .
- 202 ■ The height of  $\mathcal{T}$  is in  $O(\log n)$ , and for every node  $\mu$  of  $\mathcal{T}$ , we have  $|\text{port}(\mu)| \in O(1/\varepsilon)$ .

203 To state the final (and most important) property of  $\mathcal{T}$ , we first need to introduce some  
 204 additional notation. The properties of  $\mathcal{T}$  so far imply that the portal sets of two different  
 205 nodes in  $\mathcal{T}$  are disjoint. For every portal  $p$ , we let  $\mu(p)$  be the unique node in  $\mathcal{T}$  with  
 206  $p \in \text{port}(\mu(p))$ . Next, let  $P(s, t) = \{p \in \text{port}(\mu) \mid s, t \in V(\mu)\}$ , be the set of all portals  $p$   
 207 satisfying  $s, t \in V(\mu(p))$ . Moreover, let  $\mu$  be a node of  $\mathcal{T}$  and  $s, t \in V(\mu)$ . We denote by  
 208  $d_\mu(s, t)$  the shortest path distance between  $s$  and  $t$  in the subgraph of  $\text{DG}(V)$  induced by  
 209  $V(\mu)$ . Now, the decomposition tree of Chan and Skrepetos has the property that for every  
 210 pair of vertices  $s, t \in V$ , if we set

$$211 \quad \theta(s, t) = \min_{p \in P(s, t)} d_{\mu(p)}(s, p) + d_{\mu(p)}(p, t)$$

212 then

$$213 \quad \theta(s, t) \leq d(s, t) + O(\varepsilon D). \tag{1}$$

### 214 3.3 A Routing Scheme with Additive Stretch

215 In Section 3.1, we presented a routing scheme that is efficient for unit disk graphs with low  
 216 diameter. In this section we present a routing scheme that is efficient for unit disk graphs  
 217 with large diameter. Let  $\text{DG}(V)$  be an  $n$ -vertex unit disk graph with diameter  $D$ , and let  
 218  $\varepsilon > D^{-1}$ . We set  $c = n \cdot (\varepsilon D)^{-1}$  and define  $x_c = \lfloor x \cdot c \rfloor$ , for each  $x \in \mathbb{R}_0^+$ .

219 **The labels.** For the labels, we first compute the decomposition tree  $\mathcal{T}$ , as explained in  
 220 Section 3.2. Next, let  $v \in V$ , and let  $p$  be a portal with  $v \in V(\mu(p))$ . We compute the  
 221 shortest path tree  $T_p$  of  $V(\mu(p))$  rooted at  $p$  and enumerate its vertices in postorder. The  
 222 postorder number of  $v$  in  $T_p$  is denoted by  $r_p(v)$ . Next, the subtree of  $T_p$  rooted at  $v$  is  
 223 called  $T_p(v)$  and we use  $l_p(v)$  to denote the smallest postorder number in  $T_p(v)$ . Since we  
 224 enumerated the vertices in postorder, we get the following observation.

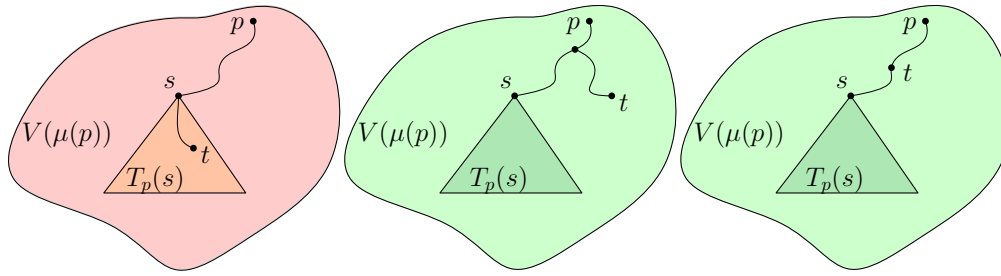
225 ► **Observation 3.5.** *Let  $w \in V(\mu(p))$ . Then we have:*

$$226 \quad w \in T_p(v) \Leftrightarrow r_p(w) \in [l_p(v), r_p(v)].$$

228 Finally, we apply the tree routing from Lemma 3.1 to  $T_p$  and denote by  $\ell_p(v)$  the corresponding  
 229 label of  $v$ . We store  $(p_{\text{id}}, d_{\mu(p)}(v, p)_c, l_p(v), r_p(v), \ell_p(v))$  in  $\ell(v)$  and get the following lemma.

230 ► **Lemma 3.6.** *For every vertex  $v \in V$ , we have  $|\ell(v)| \in O\left(\frac{\log^3 n}{\varepsilon \log \log n}\right)$ .*

231 **Proof.** Since  $\mathcal{T}$  has height  $O(\log n)$ , we know that  $v$  is in  $O(\log n)$  different sets  $V(\mu)$ .  
 232 Moreover, for every node  $\mu$ , there are at most  $O(1/\varepsilon)$  portals. Thus, the label of  $v$  contains



■ **Figure 2** Left: If  $t$  is in  $T_p(s)$ , i.e.,  $\theta(s, t; p) = d_{\mu(p)}(t, p) - d_{\mu(p)}(p, s)$ , we route away from  $p$ . Middle and Right: If  $t$  is not in  $T_p(s)$ , i.e.,  $\theta(s, t; p) = d_{\mu(p)}(t, p) + d_{\mu(p)}(p, s)$ , we route towards  $p$ . The right picture suggests to define  $\theta(s, t; p)$  as  $d_{\mu(p)}(s, p) - d_{\mu(p)}(t, p)$ . This does not influence the guarantees of our routing scheme but would lead to more cases.

233  $O(\varepsilon^{-1} \cdot \log n)$  different entries. The value  $d_{\mu(p)}(v, p)_c$  is a natural number, and since  $c \leq n$ ,  
 234 we have

235 
$$d_{\mu(p)}(v, p)_c = \lfloor d_{\mu(p)}(v, p) \cdot c \rfloor \leq n^2.$$

236 Thus, we need  $O(\log n)$  bits for the number  $d_{\mu(p)}(v, p)_c$ . Moreover, the identifier  $p_{\text{id}}$  as well  
 237 as the postorder numbers stored in one entry only need  $O(\log n)$  bits. Finally, we apply  
 238 Lemma 3.1 to conclude that one entry of the routing label has size  $O(\log^2 n / \log \log n)$ . The  
 239 claim follows. ◀

240 **The routing function.** Next, we describe the routing function. We are given the labels  $\ell(s)$   
 241 and  $\ell(t)$  for the current vertex  $s$  and the target vertex  $t$ . The dynamic part of the header  
 242 will always be empty. First, we identify all portals  $p$  with  $s, t \in V(\mu(p))$ . We can do this by  
 243 identifying all vertices  $p$  such that the entry  $(p_{\text{id}}, d_{\mu(p)}(s, p)_c, l_p(s), r_p(s), \ell_p(s))$  is in  $\ell(s)$  and  
 244 the entry  $(p_{\text{id}}, d_{\mu(p)}(t, p)_c, l_p(t), r_p(t), \ell_p(t))$  is in  $\ell(t)$ . Next, let

245 
$$\theta(s, t; p) = \begin{cases} d_{\mu(p)}(t, p) - d_{\mu(p)}(p, s), & \text{if } t \in T_p(s) \\ d_{\mu(p)}(t, p) + d_{\mu(p)}(p, s), & \text{otherwise.} \end{cases}$$

247 See Figure 2 for an illustration of the two cases. Let  $p_{\text{opt}}$  be the portal that minimizes  
 248  $\theta(s, t; p)$  among all portals  $p$ . Then, it is easy to see, that  $\theta(s, t; p_{\text{opt}}) \leq \theta(s, t)$  (recall from  
 249 Section 3.2 that  $\theta(\cdot, \cdot)$  denotes the result of the distance oracle by Chan and Skrepetos).  
 250 Hence,  $\theta(s, t; p_{\text{opt}})$  is a good approximation for the distance between  $s$  and  $t$ . However,  
 251 the routing function cannot compute the optimal portal  $p_{\text{opt}}$ , since we do not have direct  
 252 access to the real value  $d_{\mu(p)}(s, p_{\text{opt}})$ . Instead, we use the values  $d_{\mu(p)}(\cdot, p)_c$  to compute a  
 253 near-optimal portal. Analogously to  $\theta(s, t; p)$ , we define

254 
$$\theta_c(s, t; p) = \begin{cases} d_{\mu(p)}(t, p)_c - d_{\mu(p)}(p, s)_c, & \text{if } t \in T_p(s) \\ d_{\mu(p)}(t, p)_c + d_{\mu(p)}(p, s)_c, & \text{otherwise.} \end{cases}$$

256 Let  $p_0$  be the portal that lexicographically minimizes  $(\theta_c(s, t; p), p_{\text{id}})$ , among all portals  $p$ .  
 257 We call  $p_0$  the  $s$ - $t$ -portal and set  $\theta_c(s, t) = \theta_c(s, t; p_0)$ . Observe that the  $s$ - $t$ -portal can be  
 258 computed by using only the labels of  $s$  and  $t$  as well as Observation 3.5. The routing function  
 259 now uses the labels  $\ell_{p_0}(s)$  and  $\ell_{p_0}(t)$  to compute the next vertex in  $T_{p_0}$  and forwards the  
 260 data packet to this vertex (the dynamic part of the header remains empty).

## 40:8 Compact Routing in Unit Disk Graphs

261 **The stretch.** Finally, we have to show that the routing scheme is correct and routes along  
 262 a short (not necessarily shortest) path. For this, we first show that the routing process  
 263 terminates.

264 ► **Lemma 3.7.** *Let  $s$  be the current vertex,  $t$  the target vertex, and suppose that the routing  
 265 scheme sends the packet from  $s$  to  $v$ . Moreover, let  $p_0$  be the  $s$ - $t$ -portal. Then,  $p_0$  is a possible  
 266 candidate for the  $v$ - $t$ -portal, and we have  $\theta_c(s, t; p_0) \geq \theta_c(v, t; p_0) + |sv|_c$ .*

267 **Proof.** First, let  $\mu = \mu(p_0)$ . Since  $sv$  is an edge of the shortest path tree  $T_{p_0}$ , it follows that  
 268  $v \in V(\mu(p_0))$ . This gives the first part of the claim. For the second part, we distinguish two  
 269 cases:

270 **Case 1:**  $t \in T_{p_0}(s)$ . In this case, we have  $t \in T_{p_0}(v)$ , and thus  $\theta_c(v, t; p_0) = d_\mu(t, p_0)_c -$   
 271  $d_\mu(p_0, v)_c$ . Moreover, we have

$$272 \quad d_\mu(p_0, v)_c = \lfloor d_\mu(p_0, v) \cdot c \rfloor = \lfloor d_\mu(p_0, s) \cdot c + |sv| \cdot c \rfloor \geq \lfloor d_\mu(p_0, s) \cdot c \rfloor + \lfloor |sv| \cdot c \rfloor = d_\mu(p_0, s)_c + |sv|_c,$$

273 since  $s$  is on the path in  $T_{p_0}$  from  $p_0$  to  $v$ . Hence,

$$274 \quad \theta_c(s, t; p_0) = d_\mu(t, p_0)_c - d_\mu(p_0, s)_c \geq d_\mu(t, p_0)_c - d_\mu(p_0, v)_c + |sv|_c = \theta_c(v, t; p_0) + |sv|_c,$$

275 as desired.

276 **Case 2:**  $t \notin T_{p_0}(s)$ . Similarly to the first case, we have  $d_\mu(p_0, s)_c \geq d_\mu(p_0, v)_c + |sv|_c$  and  
 277  $\theta_c(v, t; p_0) \leq d_\mu(t, p_0)_c + d_\mu(p_0, v)_c$ . Thus, we get

$$278 \quad \theta_c(s, t; p_0) = d_\mu(t, p_0)_c + d_\mu(p_0, s)_c \geq d_\mu(t, p_0)_c + d_\mu(p_0, v)_c + |sv|_c \geq \theta_c(v, t; p_0) + |sv|_c,$$

279 and the claim follows. ◀

280 ► **Corollary 3.8.** *Let  $s$ ,  $t$ , and  $v$  be as in Lemma 3.7. Then,  $\theta_c(s, t) \geq \theta_c(v, t) + |sv|_c$ .*

281 **Proof.** Let  $p_0$  be the  $s$ - $t$ -portal. From Lemma 3.7, we get

$$282 \quad \theta_c(s, t) = \theta_c(s, t; p_0) \geq \theta_c(v, t; p_0) + |sv|_c \geq \theta_c(v, t) + |sv|_c. \quad \blacktriangleleft$$

283 ► **Lemma 3.9.** *Let  $s$ ,  $t$  and  $v$  be as in Lemma 3.7. Let  $p$  be the  $s$ - $t$ -portal and  $q$  be the  
 284  $v$ - $t$ -portal. Then, if  $\theta_c(s, t) = \theta_c(v, t)$ , it follows that  $p_{\text{id}} \geq q_{\text{id}}$ .*

285 **Proof.** From Lemma 3.7, we have

$$286 \quad \theta_c(v, t; q) = \theta_c(v, t) = \theta_c(s, t) = \theta_c(s, t; p) \geq \theta_c(v, t; p) + |sv|_c \geq \theta_c(v, t; p) \geq \theta(v, t; q).$$

287 Hence,  $\theta_c(v, t; p) = \theta_c(v, t; q)$ . Furthermore, by construction, we have  $(\theta_c(v, t; p), p_{\text{id}}) \geq$   
 288  $(\theta_c(v, t; q), q_{\text{id}})$ . Thus, the claim follows. ◀

289 ► **Lemma 3.10.** *The routing scheme is correct.*

290 **Proof.** Let  $s$  be the current vertex and  $t$  the desired target vertex, and let  $p$  be the  $s$ - $t$ -portal.  
 291 To measure the progress towards  $t$ , we consider the triple  $(\theta_c(s, t), p_{\text{id}}, h_p(s, t))$ , where  $h_p(s, t)$   
 292 denotes the hop distance in  $T_p$  between  $s$  and  $t$ . i.e., the number of edges on the path between  
 293  $s$  and  $t$  in  $T_p$ .

294 Suppose that the routing scheme sends the packet from  $s$  to  $v$ , and let  $q$  be the  $v$ - $t$ -  
 295 portal. We argue that  $(\theta_c(v, t), q_{\text{id}}, h_q(v, t)) < (\theta_c(s, t), p_{\text{id}}, h_p(s, t))$ . By Corollary 3.8 and



296 Lemma 3.9, it suffices to show that if  $\theta_c(s, t) = \theta_c(v, t)$  and  $p = q$ , then  $h_p(s, t) > h_q(v, t)$ .  
 297 However, this is clear, because by Lemma 3.1,  $sv$  is an edge of  $T_p$  that leads from  $s$  towards  
 298  $t$ , and  $T_q = T_p$ .

299 Now, since the triples  $(\theta_c(s, t), p_{\text{id}}, h_p(s, t))$  lie in  $\mathbb{N}^3$  and since  $(0, 0, 0)$  is a global minimum,  
 300 it follows that the data packet eventually arrives at the target vertex  $t$ . ◀

301 ▶ **Lemma 3.11.** *For any two vertices  $s$  and  $t$ , we have  $\delta(s, t) \leq d(s, t) + O(\varepsilon D)$ .*

302 **Proof.** First, we show that  $\theta_c(s, t) \leq c \cdot \theta(s, t) + 1$ : let  $p_0$  be the  $s$ - $t$ -portal, and let  $p_{\text{opt}}$  be  
 303 the portal minimizing  $\theta(s, t; \cdot)$  among all portals. Let  $\mu = \mu(p_{\text{opt}})$ . We obtain.

$$304 \quad \theta_c(s, t) = \theta_c(s, t; p_0) \leq \theta_c(s, t; p_{\text{opt}}) = \lfloor c \cdot d_\mu(t, p_{\text{opt}}) \rfloor \pm \lfloor c \cdot d_\mu(p_{\text{opt}}, s) \rfloor$$

$$305 \quad \leq \lfloor c \cdot (d_\mu(t, p_{\text{opt}}) \pm d_\mu(p_{\text{opt}}, s)) \rfloor + 1 \leq \lfloor c \cdot \theta(s, t) \rfloor + 1 \leq c \cdot \theta(s, t) + 1,$$

307 where the  $\pm$ -operator is used to cover the two possible cases in the definition of  $\theta_c$ , and  
 308 because  $\lfloor a \rfloor + \lfloor b \rfloor \leq \lfloor a + b \rfloor$  and  $\lfloor a \rfloor - \lfloor b \rfloor \leq \lfloor a - b \rfloor + 1$ , for all  $a, b \geq 0$ . By Lemma 3.10, we  
 309 know that the routing terminates. Let  $\pi : s = w_0, \dots, w_m = t$  be the routing path. From  
 310 Corollary 3.8, we get  $|w_i w_{i+1}|_c \leq \theta_c(w_i, t) - \theta_c(w_{i+1}, t)$ , and thus

$$311 \quad \delta(s, t) = \sum_{i=0}^{m-1} |w_i w_{i+1}|_c \leq \sum_{i=0}^{m-1} \frac{|w_i w_{i+1}|_c + 1}{c} = \frac{m}{c} + \frac{1}{c} \sum_{i=0}^{m-1} |w_i w_{i+1}|_c$$

$$312 \quad \leq \frac{m}{c} + \frac{1}{c} \sum_{i=0}^{m-1} (\theta_c(w_i, t) - \theta_c(w_{i+1}, t)) = \frac{m}{c} + \frac{\theta_c(s, t)}{c}$$

$$313 \quad \leq \frac{m}{c} + \frac{c \cdot \theta(s, t) + 1}{c} = \frac{m+1}{c} + \theta(s, t)$$

315 Now, using Equation (1) from Section 3.2, the choice of  $c = n \cdot (\varepsilon D)^{-1}$ , and the fact that  
 316  $m \leq n - 1$ , we get

$$317 \quad \frac{m+1}{c} + \theta(s, t) \leq \frac{n}{n \cdot (\varepsilon D)^{-1}} + d(s, t) + O(\varepsilon D) = d(s, t) + O(\varepsilon D),$$

318 as claimed. ◀

319 We can now conclude with our first theorem.

320 ▶ **Theorem 3.12.** *Let  $\text{DG}(V)$  be an  $n$ -vertex unit disk graph with diameter  $D$ . Furthermore,  
 321 let  $\varepsilon > D^{-1}$ . There is a routing scheme with label size  $O(\varepsilon^{-1} \log^3 n / \log \log n)$  whose routing  
 322 function  $\sigma_{\text{add}}$  routes any data packet on a path with additive stretch  $O(\varepsilon D)$ .*

## 323 4 A Routing Scheme with Stretch $1 + \varepsilon$

324 Let  $\text{DG}(V)$  be an  $n$ -vertex unit disk graph with diameter  $D$ , and let  $\varepsilon > 0$ . Furthermore,  
 325 without loss of generality, we can assume that  $\varepsilon \leq 1$ . For our routing scheme, we need the  
 326 following two ingredients from the literature.

327 **Planar spanners.** Let  $c \geq 1$ . A  $c$ -spanner for  $\text{DG}(V)$  is a subgraph  $H$  of  $\text{DG}(V)$  with vertex  
 328 set  $V$  such that for any  $s, t \in V$ , we have  $d_H(s, t) \leq c \cdot d(s, t)$ . The following lemma shows  
 329 the existence of good *planar* spanners for unit disk graphs and was proven by Li, Calinescu,  
 330 and Wan [16].

## 40:10 Compact Routing in Unit Disk Graphs

331 ► **Lemma 4.1.** *For any  $n$ -vertex unit disk graph  $DG(V)$ , there exists a planar 4-spanner*  
 332  *$H \subseteq DG(V)$ . The spanner  $H$  can be found in  $O(n \log n)$  time.<sup>3</sup>*

333 **Sparse covers.** Let  $H = (V, E)$  be a weighted planar graph, and let  $r \in \mathbb{N}$ . A *sparse  $r$ -cover*  
 334 *for  $H$  is a collection of connected subgraphs  $H_1, H_2, \dots$  of  $H$  with the following properties:*

335 (i) *for each vertex  $v \in V$ , there is at least one subgraph  $H_i$  that contains all the vertices*  
 336  *$w \in V$  with  $d_H(v, w) \leq r$ ;*

337 (ii) *each vertex  $v \in V$  is contained in  $O(1)$  subgraphs  $H_i$ ; and*

338 (iii)  *$\text{diam}(H_i) \leq 2^5 \cdot r$ , for every subgraph  $H_i$ , where  $\text{diam}(H_i)$  is the diameter of  $H_i$ .*

339 The following lemma establishes the existence of sparse covers for planar graphs and has  
 340 been proven by Busch, LaFortune, and Tirthapura [5].

341 ► **Lemma 4.2.** *For any weighted planar graph  $H$  with  $n$  vertices and for any  $r \in \mathbb{N}$ , we can*  
 342 *compute a sparse  $r$ -cover for  $H$  in polynomial time.*

343 **The labels.** Now we have all ingredients for our final routing scheme. We start with the  
 344 description of the labels. In the preprocessing phase, we compute a planar 4-spanner  $H$   
 345 of  $DG(V)$ , as in Lemma 4.1. Then, we have  $\text{diam}(H) \leq 4D$ . Next, for each  $k \in \mathcal{I} =$   
 346  $\{\lceil \log \frac{8}{\varepsilon} \rceil, \lceil \log \frac{8}{\varepsilon} \rceil + 1, \dots, \lceil \log(\text{diam}(H)) \rceil\}$ , we use Lemma 4.2 to construct a sparse  $2^k$ -cover  
 347  $(H_1^k, H_2^k, \dots)$  of  $H$ . Let  $G_i^k$  be the induced unit disk graph on the vertex set of  $H_i^k$ . Let  
 348  $k_0 = \lceil \log \frac{8}{\varepsilon} \rceil$ , for each  $G_i^{k_0}$ , we apply the preprocessing mechanism of the low diameter routing  
 349 scheme from Lemma 3.4. For each  $k \in \mathcal{I} \setminus \{k_0\}$ , we apply to each  $G_i^k$  the preprocessing step  
 350 of the routing scheme with additive stretch from Theorem 3.12. We use  $\ell_{k,i}$  to denote the  
 351 resulting labeling for the graph  $G_i^k$ , for  $k \in \mathcal{I}$ .

352 Now, we describe how to obtain the labels for our routing scheme. Let  $v$  be a vertex of  
 353  $DG(V)$  and let  $k \in \mathcal{I}$ . Since  $H_1^k, H_2^k, \dots$  is a sparse  $2^k$ -cover, there exists an index  $i(v, k)$   
 354 such that  $H_{i(v,k)}^k$  contains all vertices  $w \in V$  with  $d_H(v, w) \leq 2^k$ . Now, for each  $v \in V$ , the  
 355 label  $\ell(v)$  is the concatenation of the tuples  $(k, i, b(i, k, v), \ell_{k,i}(v))$ , for each  $k \in \mathcal{I}$  and each  $i$   
 356 with  $v \in V(G_i^k)$ . Here  $b(i, k, v)$  is a Boolean value that is true if and only if  $i = i(v, k)$ . The  
 357 following lemma bounds the maximum label size.

358 ► **Lemma 4.3.** *For every  $v \in V$ , we have  $|\ell(v)| \in O\left(\frac{\log D \log^3 n}{\varepsilon \log \log n} + (1/\varepsilon)^{O(\varepsilon^{-2})} \log^3 n\right)$ .*

359 **Proof.** Since there are  $O(\log D)$  different values for  $k$ , and since for each  $k$ , the vertex  $v$   
 360 appears in  $O(1)$  subgraphs  $G_i^k$ , we have that  $v$  lies in  $O(\log D)$  different subgraphs  $G_i^k$ . For  
 361 the subgraphs  $G_i^{k_0}$ , the label  $\ell_{k_0,i}(v)$  comes from the low diameter routing scheme. Since  
 362  $\text{diam}(G_i^{k_0}) \in O(1/\varepsilon)$ , Lemma 3.4 implies that  $\ell_{k_0,i}(v)$  needs  $(1/\varepsilon)^{O(\varepsilon^{-2})} \log^3 n$  bits. Since  $v$   
 363 lies in  $O(1)$  subgraphs  $G_i^{k_0}$ , we can conclude that the corresponding tuples in  $\ell(v)$  require  
 364  $(1/\varepsilon)^{O(\varepsilon^{-2})} \log^3 n$  bits in total. For the remaining  $O(\log D)$  subgraphs, we derive the label  
 365  $\ell_{k,i}(v)$  from the additive stretch routing scheme from Theorem 3.12. Hence, the corresponding  
 366 tuples take  $O(\varepsilon^{-1} \log D \log^3 n / \log \log n)$  bits in total. The claim follows. ◀

367 **The routing function.** The idea of the routing function is visualized in Figure 3. Suppose  
 368 we are given the labels  $\ell(s)$  and  $\ell(t)$  of the current vertex  $s$  and the target  $t$ , together with the  
 369 dynamic part  $h$  of the header. The routing function works as follows: We find the smallest

<sup>3</sup> Li, Calinescu, and Wan actually proved that there is a planar 2.42-spanner [16]. Since we do not care about the exact constant, we use a power of 2 to simplify later calculations.

370 number  $k = k(s, t) \in \mathcal{I}$  such that there is an index  $i$  for which the tuple  $(k, i, \text{true}, \ell_{k,i}(t))$  is  
 371 in  $\ell(t)$  and the tuple  $(k, i, *, \ell_{k,i}(s))$  is in  $\ell(s)$ .<sup>4</sup> We can now derive the following observation:

372 ► **Observation 4.4.** *Let  $s, t$  be vertices of  $G_i^k$  with  $k = k(s, t)$ . Then we have  $d(s, t) \leq 2^{k+5}$ .  
 373 Moreover, if  $k > k_0$  we have  $d(s, t) \geq 2^{k-3}$ .*

374 **Proof.** By property (iii) of a sparse cover we get  $d(s, t) \leq \text{diam}(G_i^k) \leq \text{diam}(H_i^k) \leq 2^{k+5}$ .  
 375 This proves the first inequality.

376 Next, let  $k > k_0$ . The minimality of  $k$  and property (i) of a sparse cover show that  
 377  $d_H(s, t) \geq 2^{k-1}$ . Finally, since  $H$  is a 4-spanner of  $G$  we derive  $d(s, t) \geq 2^{k-3}$  and the claim  
 378 follows. ◀

379 Once we have  $k$  and  $i$ , we can distinguish three cases.

380 If  $k > k_0$ , we ignore the dynamic header (it will be empty) and use the function  $\sigma_{\text{add}}$  of  
 381 the additive stretch routing scheme to route within  $G_i^k$ . For this, we take the labels  $\ell_{k,i}(s)$   
 382 and  $\ell_{k,i}(t)$  from  $\ell(s)$  and  $\ell(t)$  to compute the next port. The dynamic part of the header  
 383 remains empty. We use the computed port to route to the next vertex.

384 If  $k = k_0$ , we first check the dynamic header. If it is empty, we use the function  $\sigma_{\text{diam}}$  of  
 385 the low diameter routing scheme to route within  $G_i^{k_0}$ . Again, we can take the labels  $\ell_{k_0,i}(s)$   
 386 and  $\ell_{k_0,i}(t)$  from  $\ell(s)$  and  $\ell(t)$  to compute the next port. This time the routing function  
 387  $\sigma_{\text{diam}}$  also outputs a new string  $h$  of length  $O(\log^2 n / \log \log n)$ . Without loss of generality  
 388 we assume that  $h \neq \varepsilon$ . We store  $h$  in the dynamic header and route the data packet along  
 389 the computed port.

390 If  $k = k_0$  and the dynamic header contains the non-empty string  $h$ , we use  $\ell_{k_0,i}(s)$ ,  $\ell_{k_0,i}(t)$   
 391 and  $h$  to route in  $G_i^{k_0}$ , while updating the dynamic header according to  $\sigma_{\text{diam}}$ .

392 **The stretch.** It remains to show the correctness and to analyze the stretch factor. We start  
 393 with the correctness. Its proof is quite similar to the correctness proof of  $\sigma_{\text{add}}$ .

394 ► **Lemma 4.5.** *The routing scheme is correct.*

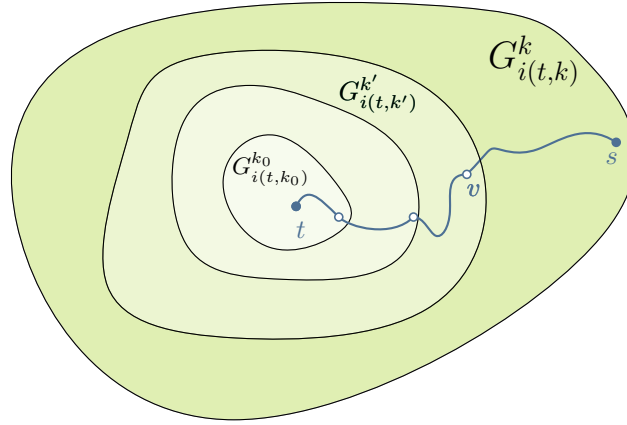
395 **Proof.** Let  $s$  be the current vertex,  $t$  the desired target vertex and suppose that the routing  
 396 scheme sends the packet from  $s$  to the vertex  $t$ . Moreover, let  $k = k(s, t)$  and  $i = i(s, t)$  be  
 397 the indices that were used by the routing function to determine  $v$ . Since the routing step  
 398 from  $s$  to  $v$  takes place in the graph  $G_i^k$ , we know that  $k$  is a potential candidate for  $k(v, t)$ .  
 399 Thus,  $k(v, t) \leq k$ . If  $k(v, t) < k$ , we have made progress. However, if  $k(v, t) = k$ , it must be  
 400 that  $i(s, t) = i(v, t)$ , since we defined  $\ell(t)$  such that for each  $k > k_0$ , there is exactly one  $i$   
 401 with  $b(i, k, t) = \text{true}$ . This means that if  $k$  does not change, the routing continues in the  
 402 subgraph  $G_i^k$ . We already proved in Lemma 3.4 and Lemma 3.10 that the underlying routing  
 403 schemes for this task are correct. Hence, after a finite number of steps, we either reach  $t$ ,  
 404 or we decrease the value  $k$ . Since there is only a finite number of values for  $k$ , correctness  
 405 follows. ◀

406 The next lemma bounds the additive stretch as a function of  $k$ .

407 ► **Lemma 4.6.** *There is a constant  $c > 0$  with the following property: let  $s$  and  $t$  be two  
 408 vertices and let  $k = k(s, t)$ . Then, we have  $\delta(s, t) \leq d(s, t) + c\varepsilon \cdot 2^k$ .*

<sup>4</sup> The  $*$  is a placeholder for an arbitrary value. Note that  $\ell(s)$  and  $\ell(t)$  each contain at most one tuple that starts with  $k, i$

40:12 Compact Routing in Unit Disk Graphs



■ **Figure 3** It holds  $2^{k-3} \leq \text{diam}(G_{i(t,k)}^k) \leq 2^{k+5}$ . We use the additive stretch routing scheme to route within  $G_{i(t,k)}^k$  until we find a vertex  $v$  that is in  $G_{i(t,k')}^{k'}$  for  $k' < k$ . This process continues until we find a vertex that is in  $G_{i(t,k_0)}^{k_0}$ , here we use the low diameter routing scheme until we reach  $t$ .

409 **Proof.** We use induction on  $k \geq k_0$ . First, suppose that  $k = k_0 = \lceil \log(8/\varepsilon) \rceil$  and let  $s, t$  be  
 410 two vertices with  $k(s, t) = k_0$ . Let  $G_i^k$  be the graph that is used to determine the next vertex  
 411 after  $s$ . Since  $k$  can only decrease while routing, and since  $k_0$  is the minimum possible value  
 412 of  $k$ , we route within  $G_i^k$ , using the low diameter routing scheme, until we reach  $t$ . Moreover,  
 413 by Lemma 3.4 and Observation 4.4, and for  $c \geq 2^5$  we get

$$414 \quad \delta(s, t) \leq (1 + \varepsilon)d(s, t) \leq d(s, t) + \varepsilon \cdot 2^{k+5} \leq d(s, t) + c\varepsilon \cdot 2^k.$$

415 Next, assume that  $k > k_0$ . Let  $s, t$  be two vertices with  $k(s, t) = k$ , and assume that for  
 416 every vertex  $w$  with  $k(w, t) < k$ , we have  $\delta(w, t) \leq d(w, t) + c\varepsilon \cdot 2^{k(w,t)}$ . Let  $G_i^k$  be the  
 417 graph in which our scheme chooses to route the data packet from  $s$  to the next node. Let  
 418  $v$  be the first node on the routing path from  $s$  to  $t$  for which  $k(v, t) < k$ , see Figure 3.  
 419 Moreover, let  $\delta'(\cdot, \cdot)$  measure the length of the routing path within the subgraph  $G_i^k$ , using  
 420 the additive stretch routing scheme. Next, by the definition of  $k_0$  and since  $k > k_0$  we  
 421 get  $\text{diam}(G_i^k) \geq d(s, t) \geq 2^{k-3} \geq 1/\varepsilon$  from Observation 4.4. Furthermore, we know that  
 422  $d(v, t) \leq \delta'(v, t)$ , since  $t$  is a vertex in  $G_i^k$ . Finally, we use the inductive hypothesis as well as  
 423 Theorem 3.12 to derive

$$424 \quad \begin{aligned} \delta(s, t) &= \delta'(s, v) + \delta(v, t) \leq \delta'(s, v) + d(v, t) + c\varepsilon \cdot 2^{k(v,t)} \leq \delta'(s, v) + \delta'(v, t) + c\varepsilon \cdot 2^{k-1} \\ &= \delta'(s, t) + c\varepsilon \cdot 2^{k-1} \leq d(s, t) + c_0\varepsilon \cdot 2^{k+5} + c\varepsilon \cdot 2^{k-1} \leq d(s, t) + c\varepsilon \cdot 2^k, \end{aligned}$$

425  
 426  
 427 for  $c \geq c_0 2^6$ , where  $c_0$  is the constant from the  $O$ -notation of the stretch in Theorem 3.12.  
 428 Hence, the claim follows. ◀

429 Finally, we can put everything together to obtain our main theorem.

430 ► **Theorem 4.7.** *Let  $\text{DG}(V)$  be an  $n$ -vertex unit disk graph and  $D$  its diameter. Furthermore,  
 431 let  $\varepsilon > 0$ . There is a routing scheme with  $(1/\varepsilon)^{O(\varepsilon^{-2})} \log D \log^3 n / \log \log n$  label size and  
 432  $O(\log^2 n / \log \log n)$  dynamic header size whose routing function achieves the stretch factor  
 433  $1 + \varepsilon$ .*

434 **Proof.** It remains to show the stretch factor. Here, it suffices to show that the stretch  
 435 factor is  $1 + O(\varepsilon)$ . Let  $s$  and  $t$  be two vertices and  $k = k(s, t)$ . If  $k = k_0$  the stretch factor

436 immediately follows from Lemma 3.4. Thus, assume  $k \neq k_0$ . On the one hand we know from  
 437 Observation 4.4 that  $2^{k-3} \leq d(s, t)$ , and on the other hand we know from Lemma 4.6 that  
 438  $\delta(s, t) \leq d(s, t) + c\varepsilon \cdot 2^k$ . Putting everything together, we get the desired stretch as follows:

$$439 \quad \delta(s, t) \leq d(s, t) + c\varepsilon \cdot 2^k \leq d(s, t) + c2^3\varepsilon \cdot d(s, t) = (1 + c2^3\varepsilon)d(s, t). \quad \blacktriangleleft$$

## 440 5 Conclusion

441 We presented an efficient and compact routing scheme for unit disk graphs. It achieves stretch  
 442  $1 + \varepsilon$  and uses  $(1/\varepsilon)^{O(\varepsilon^{-2})} \log D \log^3 n / \log \log n$  bits in the label. The dynamic header size is  
 443 bounded by  $O(\log^2 n / \log \log n)$ . It would be interesting to see if this result can be extended  
 444 to disk graphs in general. If the radii of the disks are unbounded, the decomposition of Chan  
 445 and Skrepetos cannot be applied immediately. However, the case of bounded radii is still  
 446 interesting, and even there, it is not clear how the method by Chan and Skrepetos generalizes.  
 447 If we want to decrease the size of the dynamic header and analyse the preprocessing time we  
 448 have to take a closer look into the routing scheme of Konjevod et al. [15] which we used as  
 449 blackbox.

450 Finally, let us compare our routing scheme to the known schemes. The model of the  
 451 routing scheme of Kaplan et al. [14] is very close to ours. The routing scheme can be  
 452 implemented using the fixed-port model. We achieve the same stretch factor and still use  
 453 additional information of poly-logarithmic size. Their scheme was generalized to non-unit  
 454 disk graphs with constant bounded radii [23]. Our main advantage is, that we do not use  
 455 neighborhood oracles: Kaplan et al. assumes that it can be checked locally by the routing  
 456 function (without using label, table or header) whether two vertices are neighbors or not, see  
 457 Section 5.4 in [14]. The existence of such a neighborhood oracle makes the routing much  
 458 easier, since it is a crucial problem to efficiently route in the neighborhood. However, it is  
 459 not clear how their scheme can be implemented without such an oracle.

460 The idea of the routing scheme of Yan et al. [24] is similar to ours: the graph is covered  
 461 by  $O(\log n)$  different trees. When the routing starts, the labels of the source and the target  
 462 are used to determine the identity of a tree and an  $O(\log n)$ -bit label of the target within  
 463 this tree. Finally, they completely forget the original labels and route within this tree until  
 464 they reach the target. For any two vertices  $s, t \in V$ , the routing path between  $s$  and  $t$  has  
 465 length at most  $5 \cdot d(s, t) + 13$ . Our routing scheme can also be turned into this model, but we  
 466 have  $O(\log D \log n)$  different trees that cover the unit disk graph and the label of a vertex  
 467 in one of the trees has size  $O(\log^2 n / \log \log n)$ . Nevertheless, we achieve the near optimal  
 468 stretch  $1 + \varepsilon$ . Moreover, Yan et al. use the designer-port model and thus, they can route  
 469 within a tree using labels of size  $O(\log n)$ . But since nodes are contained in more than one  
 470 tree, there have to be lookup-tables for the port assignments. Their routing scheme can  
 471 easily be turned into the fixed-port model: the stretch would not change and the label size  
 472 would increase to  $O(\log^3 n / \log \log n)$ . Last but not least, their routing scheme also achieves  
 473 constant hop stretch. It is unlikely that the hop stretch of our routing scheme is bounded by  
 474 a constant. In conclusion, our routing scheme needs an  $O(\log D)$ -factor more in the label size  
 475 but achieves a better stretch if  $\varepsilon < 4$ . Moreover, our underlying routing model is specified  
 476 more clearly.

## 477 References

- 478 1 Ittai Abraham and Cyril Gavoille. On approximate distance labels and routing schemes with  
 479 affine stretch. In *Proc. 25th Int. Symp. Dist. Comp. (DISC)*, pages 404–415, 2011.

## 40:14 Compact Routing in Unit Disk Graphs

- 480 2 Ittai Abraham, Cyril Gavoille, Andrew V. Goldberg, and Dahlia Malkhi. Routing in networks  
481 with low doubling dimension. In *26th IEEE International Conference on Distributed Computing*  
482 *Systems (ICDCS)*, page 75, 2006.
- 483 3 Baruch Awerbuch, Amotz Bar-Noy, Nathan Linial, and David Peleg. Improved routing  
484 strategies with succinct tables. *J. Algorithms*, 11(3):307–341, 1990.
- 485 4 Bahareh Banyassady, Man-Kwun Chiu, Matias Korman, Wolfgang Mulzer, André van Renssen,  
486 Marcel Roeloffzen, Paul Seiferth, Yannik Stein, Birgit Vogtenhuber, and Max Willert. Routing  
487 in polygonal domains. In *Proc. 28th Annu. Internat. Sympos. Algorithms Comput. (ISAAC)*,  
488 pages 10:1–10:13, 2017.
- 489 5 Costas Busch, Ryan LaFortune, and Srikanta Tirthapura. Sparse covers for planar graphs and  
490 graphs that exclude a fixed minor. *Algorithmica*, 69(3):658–684, 2014.
- 491 6 Timothy M. Chan and Dimitrios Skrepetos. Approximate shortest paths and distance oracles  
492 in weighted unit-disk graphs. *J. of Computational Geometry*, 10(2):3–20, 2019.
- 493 7 Shiri Chechik. Compact routing schemes with improved stretch. In *Proc. ACM Symp. Princ.*  
494 *Dist. Comp. (PODC)*, pages 33–41, 2013.
- 495 8 Brent N Clark, Charles J Colbourn, and David S Johnson. Unit disk graphs. *Discrete*  
496 *Mathematics*, 86(1-3):165–177, 1990.
- 497 9 Lenore J Cowen. Compact routing with minimum stretch. *J. Algorithms*, 38(1):170–183, 2001.
- 498 10 Tamar Eilam, Cyril Gavoille, and David Peleg. Compact routing schemes with low stretch  
499 factor. *J. Algorithms*, 46(2):97–114, 2003.
- 500 11 Pierre Fraigniaud and Cyril Gavoille. Routing in trees. In *Proc. 28th Internat. Colloq.*  
501 *Automata Lang. Program. (ICALP)*, pages 757–772, 2001.
- 502 12 Pierre Fraigniaud and Cyril Gavoille. A space lower bound for routing in trees. In *Proc. 19th*  
503 *Sympos. Theoret. Aspects Comput. Sci. (STACS)*, pages 65–75, 2002.
- 504 13 Silvia Giordano and Ivan Stojmenovic. Position based routing algorithms for ad hoc networks:  
505 A taxonomy. In *Ad hoc wireless networking*, pages 103–136. Springer-Verlag, 2004.
- 506 14 Haim Kaplan, Wolfgang Mulzer, Liam Roditty, and Paul Seiferth. Routing in unit disk graphs.  
507 *Algorithmica*, 80(3):830–848, 2018.
- 508 15 Goran Konjevod, Andréa W Richa, and Donglin Xia. Scale-free compact routing schemes in  
509 networks of low doubling dimension. *ACM Trans. Algorithms*, 12(3):1–29, 2016.
- 510 16 Xiang-Yang Li, Gruia Calinescu, and Peng-Jun Wan. Distributed construction of a planar  
511 spanner and routing for ad hoc wireless networks. In *Proceedings. Twenty-First Annual Joint*  
512 *Conference of the IEEE Computer and Communications Societies*, volume 3, pages 1268–1277.  
513 IEEE, 2002.
- 514 17 David Peleg and Eli Upfal. A trade-off between space and efficiency for routing tables. *J.*  
515 *ACM*, 36(3):510–530, 1989.
- 516 18 Liam Roditty and Roei Tov. New routing techniques and their applications. In *Proc. ACM*  
517 *Symp. Princ. Dist. Comp. (PODC)*, pages 23–32, 2015.
- 518 19 Liam Roditty and Roei Tov. Close to linear space routing schemes. *Distributed Computing*,  
519 29(1):65–74, 2016.
- 520 20 Nicola Santoro and Ramez Khatib. Labelling and implicit routing in networks. *The Computer*  
521 *Journal*, 28(1):5–8, 1985.
- 522 21 Mikkel Thorup. Compact oracles for reachability and approximate distances in planar digraphs.  
523 *J. ACM*, 51(6):993–1024, 2004.
- 524 22 Mikkel Thorup and Uri Zwick. Compact routing schemes. In *Proc. 13th ACM Symp. Par.*  
525 *Algo. Arch. (SPAA)*, pages 1–10, 2001.
- 526 23 Max Willert. Routing schemes for disk graphs and polygons. Master’s thesis, Freie Universität  
527 Berlin, 2016.
- 528 24 Chenyu Yan, Yang Xiang, and Feodor F Dragan. Compact and low delay routing labeling  
529 scheme for unit disk graphs. *Comput. Geom. Theory Appl.*, 45(7):305–325, 2012.