Headerless Routing in Unit Disk Graphs*

Wolfgang Mulzer[1] and Max Willert[1]

1 **Institut für Informatik, Freie Universität Berlin, 14195 Berlin, Germany**
{mulzer,willert}@inf.fu-berlin.de

Abstract

Let $V \subset \mathbb{R}^2$ be a set of n sites in the plane. The *unit disk graph* $\mathrm{DG}(V)$ of V is the graph with vertex set V in which 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 $\mathrm{DG}(V)$. The routing scheme $\mathcal{R}$ preprocesses $\mathrm{DG}(V)$ 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, $r = s$) and the label of the target vertex t, 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 $\mathrm{DG}(V)$, between any two sites $s, t \in V$.

We show that for any given $\varepsilon > 0$, we can construct a routing scheme for $\mathrm{DG}(V)$ with diameter D achieving stretch $1 + \varepsilon$ and label size $O(\log D \log^3 n / \log \log n)$ (the constant in the O-Notation depends on ε). In the past, several routing schemes for unit disk graphs have been proposed. Our scheme is the first one to achieve poly-logarithmic label size and arbitrarily small stretch without storing any additional information in the packet.

1 Introduction

The *routing problem* is a well-known problem in distributed graph algorithms [10, 13]. We are given a graph G and want to preprocces 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 can only use the label of the target node as well as 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 routing scheme should be *compact*: the size of the labels (in bits) must be small.

Many routing schemes use additional *headers*. The header contains mutable information and is stored in the data packet. Thus, the header moves with the data packet through the graph. The usage of an additional header makes it possible to implement recursive routing strategies or to remember information from past positions of the packet.

A trivial solution to solve the routing problem is to store the complete shortest path tree in every label. Then it is easy to route the data packets along a shortest path. However, such a routing scheme is not compact. Moreover, Peleg and Upfal [13] proved that in general graphs, any routing scheme that achieves a constant stretch factor must store a polynomial number of bits for each node.

Nevertheless, there is a rich collection of routing schemes for general graphs [1, 2, 5, 7, 8, 14, 15]. For example, the scheme by Roditty and Tov [15] uses labels of size $mn^{O(1/\sqrt{\log n})}$

* Partially supported by ERC STG 757609.

36th European Workshop on Computational Geometry, Würzburg, Germany, March 16–18, 2020.
This is an extended abstract of a presentation given at EuroCG'20. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

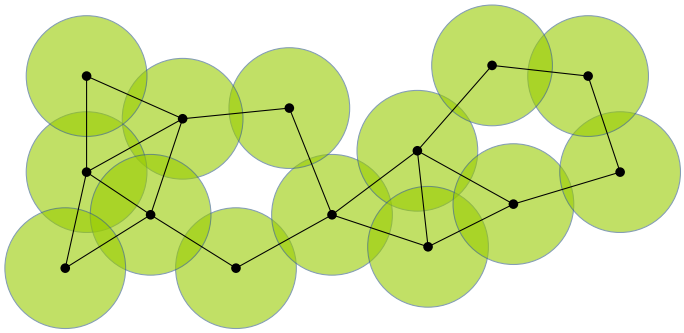

Figure 1 The disks in the unit disk graph have diameter 1 and there is an edge between two midpoints if and only if their corresponding disks intersect.

and routes a packet from s to t on a path of length $O\big(k\Delta + m^{1/k}\big)$, where Δ is the shortest path distance between s and t, $k > 2$ is any fixed integer, n is the number of nodes, and m is the number of edges. Their routing scheme needs headers of poly-logarithmic size. There are routing schemes for special graph classes that achieve better bounds on the label size and stretch factor [3, 9, 16–18].

Our graph class of interest comes from the study of mobile and wireless networks. These networks are usually modeled as *unit disk graphs* [6] with diameter D. There are already several routing schemes for unit disk graphs [11, 20]. We present the first headerless routing scheme with label size $O(\varepsilon^{-4} \log D \log^3 n / \log\log n)$ that achieves stretch $1 + \varepsilon$.

2 Preliminaries

Let $G = (V, E)$ be a *simple*, *undirected*, and *connected* graph with n vertices. In our model the graph G is embedded in the Euclidean plane and an edge uv is weighted according to the Euclidean distance $|uv|$ of its endpoints, for all , $uv \in V$. We write $d(u, v)$ for the (weighted) *shortest path distance* between the vertices $u, v \in V$. Moreover, every vertex v has a unique identifier $v_{\text{id}} \in \{0, \ldots, n-1\}$. In a *unit disk graph* $\mathrm{DG}(V)$ of V, there is an edge between two nodes $u, v \in V$ if and only if $|uv| \leq 1$, see Figure 1. We use D to denote the diameter $\max_{u,v \in V} d(u, v)$ of $\mathrm{DG}(V)$. We assume that $\mathrm{DG}(V)$ is connected. Hence, $D \leq n - 1$.

Routing Schemes. Let $G = (V, E)$ be a graph. A *routing scheme* $\mathcal{R}$ for G consists of a *labeling function* $\ell(v) \in \{0, 1\}^+$. It serves as the address of the node v in G and might contain some more information about the topology of G. Furthermore, $\mathcal{R}$ has a *routing function* $\sigma : \ell(V) \times \ell(V) \to V$. The routing function σ describes the behavior of the routing scheme, as follows: assume a data packet is located at a vertex $s \in V$ and must be routed to a destination $t \in V$. Then, $\sigma(\ell(s), \ell(t))$ has to compute a vertex to which the data packet is forwarded. Now, let $v_0 = s$ and $v_{i+1} = \sigma(\ell(v_i), \ell(t))$, for $i \geq 0$. The sequence $(v_i)_{i \in \mathbb{N}}$ is called *routing sequence*. The routing scheme $\mathcal{R}$ is *correct*, if and only if for all distinct $s, t \in V$, there is a number $m \in \mathbb{N}$ such that $v_j = t$, for all $j \geq m$, and $v_j \neq t$, for all $j = 0, \ldots, m-1$. If $\mathcal{R}$ is correct for G, then $\delta(s, t) = \sum_{i=1}^{m} |v_{i-1} v_i|$ is called the *routing length* between s and t (in G). The *stretch* of the routing scheme is the largest ratio $\delta(s, t)/d(s, t)$ 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.

3 Building Blocks

The Distance Oracle of Chan and Skrepetos. Our routing scheme is based on the recent approximate distance oracle for unit disk graphs by Chan and Skrepetos [4]: we are given an n-vertex unit disk graph with diameter D and a parameter $\varepsilon \geq D^{-1}$. Chan and Skrepetos show how to compute an efficient data structure that can answer *approximate distance queries* in $\mathrm{DG}(V)$: given two vertices $s, t \in V$, compute a 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 *decomposition tree* $\mathcal{T}$ for $\mathrm{DG}(V)$ with the following properties.

- Every node μ of $\mathcal{T}$ is assigned two sets $\mathrm{port}(\mu)$ and $V(\mu)$ such that $\mathrm{port}(\mu) \subseteq V(\mu) \subseteq V$. The subgraph of $\mathrm{DG}(V)$ induced by $V(\mu)$ is connected and the vertices in $\mathrm{port}(\mu)$ are called *portals*.
- If μ is the root, then $V(\mu) = V$. If μ is a leaf, then $V(\mu) = \mathrm{port}(\mu)$.
- If μ is an inner node with k children $\sigma_1, \ldots, \sigma_k$, the sets $\mathrm{port}(\mu), V(\sigma_1), \ldots, V(\sigma_k)$ are pairwise disjoint, and we have $V(\sigma_i) \subseteq V(\mu)$, for $1 \leq i \leq k$.
- The height of $\mathcal{T}$ is in $O(\log n)$, and for every node μ of $\mathcal{T}$, we have $|\mathrm{port}(\mu)| \in O(1/\varepsilon)$.

To state the final (and most important) property of $\mathcal{T}$, we need some additional notation. The properties of $\mathcal{T}$ so far imply that the portal sets of two different nodes in $\mathcal{T}$ are disjoint. For every portal p, we let $\mu(p)$ be the unique node in $\mathcal{T}$ with $p \in \mathrm{port}(\mu(p))$. Moreover, let μ be a node of $\mathcal{T}$ and $s, t \in V(\mu)$. We denote by $d_\mu(s,t)$ the shortest path distance between s and t in the subgraph of $\mathrm{DG}(V)$ induced by $V(\mu)$. Now, the decomposition tree of Chan and Skrepetos has the property that for every pair of vertices $s, t \in V$, if we set

$$\theta(s,t) = \min_{\substack{p \text{ portal} \\ s,t \in V(\mu(p))}} d_{\mu(p)}(s,p) + d_{\mu(p)}(p,t)$$

then

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

Simple Routing Schemes. Moreover, we need the following known routing schemes. The first routing scheme is due to Fraigniaud and Gavoille [9] as well as Thorup and Zwick [18]. The second routing scheme is based on an idea described by Kaplan et al. [11]. For the proof, we refer to [12].

▶ **Lemma 1.** *Let T be an n-vertex tree with arbitrary edge weights. There is a routing scheme for T with label size $O(\log^2 n / \log\log n)$ and stretch 1.*

▶ **Lemma 2.** *Let $\mathrm{DG}(V)$ be an n-vertex unit disk graph with diameter D and $0 < \varepsilon \leq D^{-1}$. There is a routing scheme for $\mathrm{DG}(V)$ with label size $O\left(\varepsilon^{-4} \log n\right)$ and stretch $1 + O(\varepsilon)$.*

4 A Routing Scheme with Additive Stretch

In this section we present a routing scheme that is efficient for $\mathrm{DG}(V)$ if the diameter D is large. Let $\varepsilon > D^{-1}$ and $c = n \cdot (\varepsilon D)^{-1}$. We define $x_c = \lfloor x \cdot c \rfloor$, for each $x \in \mathbb{R}_0^+$. Let $\mathcal{T}$ be the decomposition tree of $\mathrm{DG}(V)$, as explained in Section 3.

The idea of the routing scheme is as follows: We use $\mathcal{T}$ to compute a set of shortest path trees whose union covers $\mathrm{DG}(V)$ such that every vertex is contained in at most $O(\varepsilon^{-1} \log n)$ trees. In each step of the routing process, we use the source and target labels to select a good shortest path tree and route in this tree.

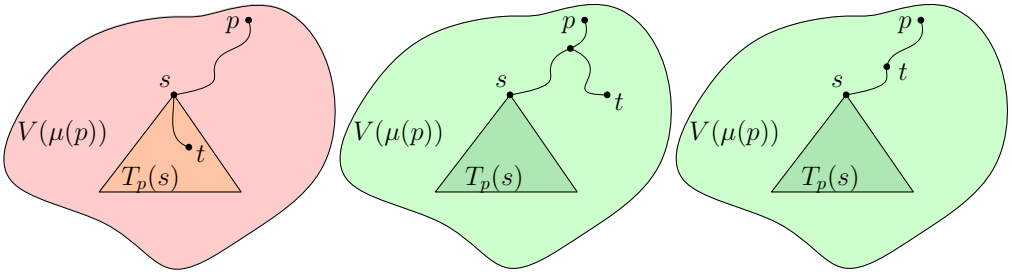

Figure 2 Left: If t is in $T_p(s)$, i.e., $\theta(s,t;p) = d_{\mu(p)}(t,p) - d_{\mu(p)}(p,s)$, we route away from p. Middle and Right: If t is not in $T_p(s)$, i.e., $\theta(s,t;p) = d_{\mu(p)}(t,p) + d_{\mu(p)}(p,s)$, we route towards p. The right picture suggests to define $\theta(s,t;p)$ as $d_{\mu(p)}(s,p)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

The Labels. Let $v \in V$, and let p be a portal with $v \in V(\mu(p))$. We compute the shortest path tree T_p for p in $V(\mu(p))$ and enumerate its vertices in postorder. The postorder number of v in T_p is denoted by $r_p(v)$. Next, the subtree of T_p rooted at v is called $T_p(v)$ and we use $l_p(v)$ to denote the smallest postorder number in $T_p(v)$. Thus, a vertex $w \in V(\mu(p))$ is in the subtree $T_p(v)$ if and only if $r_p(w) \in [l_p(v), r_p(v)]$. Finally, we apply the tree routing from Lemma 1 to T_p and denote by $\ell_p(v)$ the corresponding label of v. We store $(p_{\mathrm{id}}, d_{\mu(p)}(v,p)_c, l_p(v), r_p(v), \ell_p(v))$ in $\ell(v)$, for every portal p. The rounding of the distances is necessary since we are not allowed to store real values. For the proof of the next lemma see [12].

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

The Routing Function. We are given the labels $\ell(s)$ and $\ell(t)$ for the current vertex s and the target vertex t. First, we identify all portals p with $s,t \in V(\mu(p))$. We can do this by identifying all vertices p such that the entry $(p_{\mathrm{id}}, d_{\mu(p)}(s,p)_c, l_p(s), r_p(s), \ell_p(s))$ is in $\ell(s)$ and the entry $(p_{\mathrm{id}}, d_{\mu(p)}(t,p)_c, l_p(t), r_p(t), \ell_p(t))$ is in $\ell(t)$. Next, let $\theta(s,t;p) = d_{\mu(p)}(t,p) + d_{\mu(p)}(p,s)$, if t is not in the subtree $T_p(s)$, and $\theta(s,t;p) = d_{\mu(p)}(t,p) - d_{\mu(p)}(p,s)$, otherwise; see Figure 2 for an illustration of the two cases. Let p_{opt} be the portal that minimizes $\theta(s,t;p)$ among all portals p. Then, it is easy to see, that $\theta(s,t;p_{opt}) \leq \theta(s,t)$. Hence, $\theta(s,t;p_{opt})$ is a good approximation for the distance between s and t and we would like to route in $T_{p_{opt}}$. However, the routing function cannot compute the optimal portal p_{opt}, since we do not have direct access to the exact distances. Instead, we use the rounded distances to compute a near-optimal portal. We define $\theta_c(s,t;p) = d_{\mu(p)}(t,p)_c + d_{\mu(p)}(p,s)_c$, if t is not in the subtree $T_p(s)$, and $\theta_c(s,t;p) = d_{\mu(p)}(t,p)_c - d_{\mu(p)}(p,s)_c$, otherwise. Let p^* be the portal that lexicographically minimizes $(\theta_c(s,t;p), p_{\mathrm{id}})$, among all portals p. We call p^* the *s-t-portal* and set $\theta_c(s,t) = \theta_c(s,t;p^*)$. Observe that the s-t-portal can be computed by using only the labels of s and t. The routing function now uses the labels $\ell_{p^*}(s)$ and $\ell_{p^*}(t)$ to compute the next vertex in T_{p^*} and forwards the data packet to this vertex.

Analysis. The following lemma shows that we make progress after each step.

▶ **Lemma 4.** *Let s be the current vertex, t the target vertex, and suppose that the routing scheme sends the packet from s to v. Moreover, let p be the s-t-portal and q be the v-t-portal. We have*

1. $\theta_c(s,t) \geq \theta_c(v,t) + |sv|_c$,
2. *if* $\theta_c(s,t) = \theta_c(v,t)$ *then* $p_{\mathrm{id}} \geq q_{\mathrm{id}}$, *and*
3. *if* $\theta_c(s,t) = \theta_c(v,t)$ *and* $p_{\mathrm{id}} = q_{\mathrm{id}}$ *then* v *is on a shortest path from* s *to* t *in* T_q.

The intuition is as follows. First of all, the rounded approximate distance to the target will never increase (1st statement). If this values does not change, then the next tree in which we route can not have a larger index (2nd statement). If this index does not change, then we decrease the hop distance to our target (3rd statement). Hence, we made progress. The first statement of Lemma 4 can now be used to obtain the stretch factor and therefore the main theorem. The proof uses Inequality 1 and can be found in [12].

▶ **Theorem 5.** *Let* $\mathrm{DG}(V)$ *be an* n*-vertex unit disk graph with diameter* D *and* $\varepsilon > D^{-1}$. *There is a routing scheme for* $\mathrm{DG}(V)$ *with label size* $O\left(\varepsilon^{-1} \log^3 n / \log\log n\right)$ *and additive stretch* $O(\varepsilon D)$.

Theorem 5 and Lemma 2 can now be used as building blocks to get a routing scheme with stretch factor $1+\varepsilon$. To achieve this we use a well-known technique that groups the vertices of $\mathrm{DG}(V)$ using a hierarchy of sparse covers with exponentially increasing diameter [4]. In the end, each node is contained in at most $O(\log D)$ different groups. Each group gives a connected subgraph of $\mathrm{DG}(V)$ on which we apply one of the two routing schemes, depending on whether $\varepsilon \geq D^{-1}$ or not. It is then easy to route in these subgraphs. For the details and the analysis we refer to [12]. Nevertheless, we claim the following.

▶ **Theorem 6.** *Let* $\mathrm{DG}(V)$ *be an* n*-vertex unit disk graph with diameter* D *and* $\varepsilon > 0$. *There is a routing scheme for* $\mathrm{DG}(V)$ *with label size* $O\left(\varepsilon^{-4} \log D \log^3 n / \log\log n\right)$ *and stretch* $1+\varepsilon$.

5 Conclusion

We presented an efficient, compact, and headerless routing scheme for unit disk graphs. It achieves near-optimal stretch $1+\varepsilon$ and uses $O(\log D \log^3 n / \log\log n)$ bits in the label.

It would be interesting to see if this result can be extended to disk graphs in general. If the radii of the disks are unbounded, the decomposition of Chan and Skrepetos cannot be applied immediately. However, the case of bounded radii is still interesting, and even there, it is not clear how the method by Chan and Skrepetos generalizes.

Finally, let us compare our routing scheme to the known schemes. The model of the routing scheme of Kaplan et al. [11] is very close to ours. They claim that the neighborhood can be checked locally without wasting storage. We also use this assumption in the details of Lemma 2. The scheme was generalized to non-unit disk graphs with constant bounded radii [19]. Nevertheless, in unit disk graphs, we achieve the same stretch factor and still have additional information of poly-logarithmic size. The main advantage of our routing scheme is that we do not use any additional headers. Therefore, whenever a data packet arrives at a node, it is not necessary to know what happened before or where the packet came from. In the routing scheme of Kaplan et al., a data packet visits a node more than once.

The routing scheme of Yan et al. [20] uses headers as well, but they are only computed in the first step and do not change again. The idea of their routing scheme is similar to ours: the graph is covered by $O(\log n)$ different trees. When the routing starts, the labels of the source and the target are used to determine the identity of a tree and an $O(\log n)$-bit label of the target within this tree. Finally, they completely forget the original labels and route within this tree until they reach t. Their stretch is bounded by a constant. Our routing scheme can also be turned into this model, but we have $O(\log D \log n)$ different trees that cover the

unit disk graph and the label of a vertex in one of the trees has size $O(\log^2 n/\log\log n)$. Nevertheless, we achieve the near optimal stretch $1+\varepsilon$. A more thorough analysis of their and our model will lead to a more complicated comparison. This does not fit here, so we refer the reader to have a look into [12].

References

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

2 Baruch Awerbuch, Amotz Bar-Noy, Nathan Linial, and David Peleg. Improved routing strategies with succinct tables. *J. Algorithms*, 11(3):307–341, 1990.

3 Bahareh Banyassady, Man-Kwun Chiu, Matias Korman, Wolfgang Mulzer, André van Renssen, Marcel Roeloffzen, Paul Seiferth, Yannik Stein, Birgit Vogtenhuber, and Max Willert. Routing in polygonal domains. In *LIPIcs-Leibniz International Proceedings in Informatics*, volume 92. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2017.

4 Timothy M. Chan and Dimitrios Skrepetos. Approximate shortest paths and distance oracles in weighted unit-disk graphs. *J. of Computational Geometry*, 10(2):3–20, 2019. doi:10.20382/jocg.v10i2a2.

5 Shiri Chechik. Compact routing schemes with improved stretch. In *Proc. ACM Symp. Princ. Dist. Comp. (PODC)*, pages 33–41, 2013.

6 Brent N Clark, Charles J Colbourn, and David S Johnson. Unit disk graphs. *Discrete mathematics*, 86(1-3):165–177, 1990.

7 Lenore J Cowen. Compact routing with minimum stretch. *J. Algorithms*, 38(1):170–183, 2001.

8 Tamar Eilam, Cyril Gavoille, and David Peleg. Compact routing schemes with low stretch factor. *J. Algorithms*, 46(2):97–114, 2003.

9 Pierre Fraigniaud and Cyril Gavoille. Routing in trees. In *Proc. 28th Internat. Colloq. Automata Lang. Program. (ICALP)*, pages 757–772, 2001.

10 Silvia Giordano and Ivan Stojmenovic. Position based routing algorithms for ad hoc networks: A taxonomy. In *Ad hoc wireless networking*, pages 103–136. Springer-Verlag, 2004.

11 Haim Kaplan, Wolfgang Mulzer, Liam Roditty, and Paul Seiferth. Routing in unit disk graphs. *Algorithmica*, 80(3):830–848, 2018.

12 Wolfgang Mulzer and Max Willert. Routing in unit disk graphs without dynamic headers, 2020. arXiv:2002.10841.

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

14 Liam Roditty and Roei Tov. New routing techniques and their applications. In *Proc. ACM Symp. Princ. Dist. Comp. (PODC)*, pages 23–32, 2015.

15 Liam Roditty and Roei Tov. Close to linear space routing schemes. *Distributed Computing*, 29(1):65–74, 2016.

16 Nicola Santoro and Ramez Khatib. Labelling and implicit routing in networks. *The Computer Journal*, 28(1):5–8, 1985.

17 Mikkel Thorup. Compact oracles for reachability and approximate distances in planar digraphs. *J. ACM*, 51(6):993–1024, 2004.

18 Mikkel Thorup and Uri Zwick. Compact routing schemes. In *Proc. 13th ACM Symp. Par. Algo. Arch. (SPAA)*, pages 1–10, 2001.

19 Max Willert. Routing schemes for disk graphs and polygons. Master's thesis, Freie Universität Berlin, 2016.

20 Chenyu Yan, Yang Xiang, and Feodor F Dragan. Compact and low delay routing labeling scheme for unit disk graphs. *Comput. Geom. Theory Appl.*, 45(7):305–325, 2012.