

# On Complexity of Wireless Gathering Problems on Unit-Disk Graphs

Nikola Milosavljević \*

Max-Planck-Institut für Informatik, Saarbrücken, Germany  
nikolam@mpi-inf.mpg.de

**Abstract.** We address the problem of efficient gathering of data from a wireless network to a single sink node. Network’s communication and interference pattern are assumed to be captured by the unit-disk model. We consider the objective of minimizing the maximum latency (the time between release and delivery of a packet). We prove that the problem is NP-complete even when all packets are released from the same source node, or at the same time. To our knowledge, these are the first results about the wireless gathering problem in the plane. They can be seen as an extension of recent inapproximability results of Bonifaci *et al.*, which hold in three dimensions.

**Keywords:** wireless gathering problem, unit-disk graphs, wireless networks, computational complexity

## 1 Introduction

Wireless networks have become ubiquitous in recent years. Some examples include 802.11 wireless Internet, mobile networks for voice and data communication, vehicular networks, sensor/actuator networks which measure physical data in a given environment (natural habitat, office building), and act to achieve desired environmental effect (energy savings, event notification etc.)

One of the most important communication tasks in all these applications is *gathering data* from some or all nodes (*data sources*) into one distinguished node (*sink*). For example, in an 802.11 network, the sources are individual WiFi transceivers and the sink is a WiFi access point. In a sensor/actuator network, the sources are sensor nodes and the sink is a more powerful “hub” node, a base station with human operator that examines the data, or a control unit that turns sensor data into actuators’ control signals.

In most wireless networks data is being generated asynchronously in real time. New data packets arrive as others are being relayed. In this setting, a natural performance metric for a data gathering scheme is *flow time* or *latency*: the time that a packet spends in the network, between the time it is generated at a source, and the time it is delivered to the sink. In this paper we consider

---

\* The author would like to thank anonymous reviewers for suggesting parts of the original submission that needed extra clarification.

the objectives related to flow time: maximum flow time of any packet, and total flow time of all packets.

The feature that distinguishes scheduling problems in wireless networks from those in wired networks is the nature of interference. In wireless networks, transmitting a packet along a link does not only affect the endpoints of that link, but also all nodes in radio range of the sender, regardless of whether they are intended recipients or not. Different models of radio propagation yield different interference patterns, resulting in optimization problems with different theoretical properties. In this paper we adopt the *unit-disk model* of radio propagation, under which a node’s radio range is a disk of radius 1 centered at the node. In other words, the network is modeled by a *unit-disk graph*, in which two nodes are connected if and only if their Euclidean distance is at most 1. Although far from realistic, the unit-disk model is widely used in the literature and generally accepted as a good first approximation.

### 1.1 Wireless Gathering Problem

We now formally define the decision problems that are the topic of this paper, and introduce the associated terminology. The input is a set of points in the plane (given by their Cartesian coordinates), representing *nodes* of the wireless network. One node is designated as the *sink*. We assume that the time consists of discrete steps. In the beginning of each step (before any transmissions take place), a number of packets is generated (released) at some nodes in the network (*sources*). The release pattern of the packets is also given as input. Each node can transmit at most one packet (message) in one step, and that packet must be addressed to exactly one other node (intended recipient)<sup>1</sup>. A transmission from node  $u$  to node  $v$  is received successfully by  $v$  if and only if the only node within distance 1 from  $v$  which is transmitting in that step is  $u$ . The goal is to schedule transmissions so that all packets are eventually delivered to the sink. Given such a schedule, a packet’s *latency* is the difference between its release time and its delivery time. In this paper, *wireless gathering problem* is the problem of computing a schedule that minimizes *maximum latency* over all packets. We consider two special cases:

- when all packets are generated at the same source node, possibly at different times,
- when all packets are generated at the same time (without loss of generality at time zero), possibly at different nodes.

We denote these by T-WG and S-WG, respectively<sup>2</sup>. Note that S-WG is equivalent to minimizing the *completion time*, i.e., total length of the schedule.

---

<sup>1</sup> Note that even though “local broadcast” (simultaneous transmission from one sender to multiple receivers) is supported by the wireless medium, and often helpful in real-world scenarios, our model does not allow it.

<sup>2</sup> Prefixes T- and S- refer to the quantities that *differ* among packets, i.e., “time” and “source”, respectively. WG stands for ‘wireless gathering’.

## 1.2 Related Work

The literature on routing and scheduling in wireless networks is extremely large. Existing work can be classified according to interference model (geometric, graph-based etc.), objective function (completion time, maximum latency, average latency etc.)... We only mention publications most relevant to our results.

Kumar *et al.* [5] study the problem of routing packets between *different source-destination pairs* in minimum number of steps under *distance-2 interference model*. They prove strong inapproximability for general graphs and give approximation algorithms for disk graphs.

Bermond *et al.* [2] consider the problem of collecting data to a *single sink node* in minimum number of steps. Their interference model is defined by transmission radius  $r_T$  and interference radius  $r_I$ , which are the same for each node, and measured in network's hop-distance, rather than geometric distance. They provide a 4-approximation algorithm when  $r_I > r_T$ , and prove that the problem is NP-hard for  $r_I = r_T$ .

Balakrishnan *et al.* [1] considered the problem of maximizing the number of transmissions that can be scheduled simultaneously. They prove that the problem is NP-complete even in the case of planar unit-disk graphs. The proof is based on the fact that in a bipartite graph, a set of transmissions can be scheduled simultaneously if and only if the corresponding set of edges (sender-receiver pairs) forms an induced matching (IM)<sup>3</sup> (IM for short). Hardness then follows by reduction from IM on a certain subclass of bipartite graphs which also happen to be planar unit-disk graphs.

Bonifaci *et al.* [3] consider the case when the nodes lie in the 3D space, and communication/interference pattern is given by the *unit-ball* model, the obvious three-dimensional analogue of the unit-disk model. They prove that unless  $P=NP$ , T-WG cannot be  $O(p^{1-\varepsilon})$ -approximated in polynomial time for any  $\varepsilon > 0$ , where  $p$  is the number of packets. They also prove inapproximability of minimizing the *average* latency. They give a constant approximation algorithm for S-WG, as well as resource-augmented algorithms for T-WG (both maximum and average latency objective).

Proofs of inapproximability in [3] also exploit hardness of IM on a subclass of bipartite graphs similar to that in [1]. The idea is to connect one half of the bipartition to the source, and the other half to the sink. That way, the most efficient way of gathering data is clearly to find the largest set of simultaneously schedulable links in the bipartite graph and use it repeatedly to transfer all packets. Unfortunately, it seems that adding these extra links really requires using the third dimension. The third dimension is used to separate the two halves of the bipartition in space. Once this separation is achieved, one can place

---

<sup>3</sup> A *matching* in an undirected graph is a subgraph in which no two edges share an endpoint. An *induced matching* is a matching that is induced by a subset  $S$  of vertices, i.e., consists of  $S$  and all edges connecting vertices in  $S$ . The problem of deciding if a graph has an IM of given size (denoted by IM in the rest of the paper) is NP-hard even for graphs that are simultaneously planar, bipartite, unit-disk, and have maximum degree at most 4 [7, 1].

source and sink so that they are connected only to one half of the bipartition. The main technical contribution of this paper is showing that if we settle for “weaker” connection between the bipartite graph and the source/sink (not by direct links, but longer paths), then two dimensions are enough to prove “weaker” hardness.

### 1.3 Our Results

We prove that both T-WG and S-WG are NP-complete. Our proofs are by reduction from IM on a subclass of planar bipartite unit-disk graphs similar to that in [1, 3]. For the graphs in this class we propose a unit-disk embedding in which the two halves of the bipartitions are spatially separated well enough that they can be connected to the source/sink by *paths*, while still respecting the unit-disk constraint.

The rest of the paper is organized as follows. Section 2 contains the proofs of our two claims. Section 3 summarizes the paper and suggests some problems for further research.

## 2 NP-Completeness of Wireless Gathering Problems

We prove NP-completeness of T-WG and S-WG by reduction from IM on a subclass of bipartite unit-disk planar graphs which we call *grid graphs*<sup>4</sup>. In Section 2.1 we define grid graphs, and argue that IM is indeed NP-hard when restricted to this class.

Then we proceed to describe the reductions to T-WG and S-WG. The network (set of points) in the output of both reductions is the same, only the packet release patterns are different. The network is obtained in three stages:

- (i) embedding the grid graph onto a narrow strip, so that the two sides of the bipartition are well separated along the strip (Section 2.2),
- (ii) embedding several copies of the result onto a ring (annulus), so that the two sides of the bipartition are well separated along the ring (Section 2.3),
- (iii) inserting additional nodes to induce paths that connect the ring embedding to the source and sink, placed inside and outside of the ring, respectively (Section 2.4).

In Sections 2.5 and 2.6 we describe the packet release patterns for T-WG and S-WG, respectively, and prove correctness of respective reductions.

### 2.1 Grid Graphs

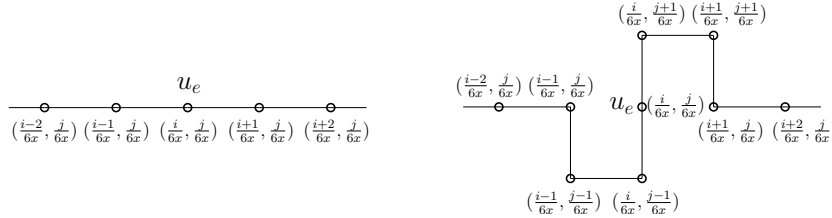
Let  $G$  be a cubic graph (planar graph with maximum degree 3) with  $n$  vertices and  $m$  edges. Let  $a, b$  be arbitrary positive integers, and let  $\delta > 0$  be arbitrarily small real number.

<sup>4</sup> We borrow this term from Bonifaci *et al.* [3], although our definition is slightly different.

**Lemma 1.** Given  $G$ ,  $a$ ,  $b$ ,  $\delta$ , one can compute in polynomial time a graph  $H$  such that the following holds.

- (i)  $H$  can be embedded as a unit-disk graph on some finite subset of the integer grid  $\mathbb{Z}^2$ . We refer to this embedding as the grid embedding.
- (ii)  $H$  is a subdivision of  $G$ ; edge  $e$  of  $G$  is replaced by a path of length  $6a_e - 2$ , where  $a_e \geq a$ .
- (iii) The length of any horizontal or vertical path in the grid embedding of  $H$  is even, and at least  $b$ .
- (iv) The ratio of the number of horizontal and vertical edges is at most  $\delta$ .

*Proof.* Compute a drawing of  $G$  which is *planar* (no two edges intersect), *orthogonal* (edges are polygonal chains with alternating horizontal and vertical segments) and *grid* (vertices of the graph and the polygonal chains have integer coordinates). Such a drawing can be computed in polynomial time [6, 8]. Clearly, each edge  $e$  has integer length  $z_e$ . Choose  $x \in \mathbb{Z}^+$  large enough so that the following holds: each edge  $e$  in the drawing has a point  $u_e = (\frac{i}{6x}, \frac{j}{6x})$ ,  $i, j \in \mathbb{Z}$  in its interior such that the drawing in the disk of radius  $\frac{1}{2x}$  centered at  $u_e$  is just a horizontal or vertical line across the diameter of the disk. On each edge replace a straight line segment of length  $\frac{1}{3x}$  centered at  $u_e$  by a polygonal chain of length  $\frac{1}{x}$  without introducing edge crossings, as shown in Figure 1 below. This is possible by choice of  $x$ . The length of edge  $e$  is now  $z_e + \frac{2}{3x}$ , and the length of any horizontal or vertical segment is at least  $\frac{1}{6x}$ . Scale up the drawing by a factor of  $6x(3y + 1)$ , where  $y$  is a fixed integer. The length of edge  $e$  is now  $6[(3y + 1)xz_e + (2y + 1)] - 2$ , and the length of any horizontal or vertical segment is at least  $3y + 1$ . Subdivide  $G$  by adding all vertices with integer coordinates



**Fig. 1.** Proof of Lemma 1: one step in obtaining a grid graph from a cubic graph.

that lie in the interiors of edges in the drawing. We claim that the resulting subdivision of  $G$  is in fact desired  $H$ . The drawing inherited from  $G$  is clearly a grid embedding. Edge  $e$  is subdivided  $6a_e - 2$  times, where  $a_e = (3y + 1)xz_e + (2y + 1)$ . The length of any horizontal or vertical path is  $3y + 1$ . One can ensure  $a_e \geq a$  and  $3y + 1 \geq b$  by choosing  $y$  to be a large odd integer.

Finally, one can ensure (iv) as follows. Find a row in the grid that contains only “simple” vertical edges, i.e., those that are not involved in turns, junctions etc. Such a row exists by previous construction; in particular, for large enough

y. Insert below this row a number of its copies, which should be divisible by 6, so as not to affect (i)–(iii). Clearly, this only adds vertical edges, so (iv) holds for large enough number of copies.  $\square$

**Definition 1.** A grid graph is any graph  $H_2$  obtained from a cubic graph  $G$  by first computing  $H_1$  as in Lemma 1, and then modifying  $H_1$  as follows: for each vertex  $v$  of  $H_1$  which corresponds to a vertex of  $G$  (i.e., is not a subdivision vertex), create a new vertex  $v'$  and an edge  $(v, v')$  in  $H_2$ .

Since  $H_1$  in Lemma 1 has maximum degree 3, a vertex  $v'$  can always be added safely as the fourth neighbor of  $v$  in the grid. Furthermore, vertices  $v'$ , thus placed, cannot “collide” with each other in the embedding, because they are added only for vertices  $v$  that are far away from each other, by property (iii) of Lemma 1. We conclude that  $H_2$  satisfies property (i) of Lemma 1. In particular, grid graphs are bipartite.

We end this section by proving that IM is NP-complete on grid graphs, by reduction from the independent set problem on cubic graphs. The reduction is given by Definition 1, and its correctness follows from Lemma 2 below. The proof of Lemma 2 is very similar to those in [7] (Theorem 2.1) and [1] (Section 11.1). For completeness, we reproduce it here with necessary minor modifications.

**Lemma 2.**  $H_2$  has an IM of size  $2\sum_e a_e - m + \alpha$  if and only if  $G$  has an independent set of size  $\alpha$ .

*Proof.* If  $G$  has an independent set  $I$  of size  $\alpha$  or more, then one can obtain an IM  $S$  of size  $2\sum_e a_e - m + \alpha$  as follows. For each  $v \in I$ , put  $(v, v')$  into  $S$ . For each edge  $e = (u, v)$  of  $G$ , do the following. Assume without loss of generality that  $u \notin I$ . Consider the path  $u = w_1, w_2, \dots, w_{6a_e-1} = v$  in  $H_2$  which is the subdivision of  $e$ . Put into  $S$  edges  $(w_{3i-1}, w_{3i})$ ,  $i = 1, 2, \dots, 2a_e - 1$ .

Now suppose that there is an IM  $S$  of size at least  $2\sum_e a_e - m + \alpha$ . Let  $S_e$  be a subset of  $S$  that belongs to the subdivision of edge  $e$ . Suppose that  $|S_e| < 2a_e - 1$  for some  $e = (u, v)$ . Let  $u = w_1, w_2, \dots, w_{6a_e-1} = v$  be the nodes in the subdivision. Let  $(u, w)$  be the only edge of  $S$  adjacent to  $u$  (if any). Replace  $S$  by  $S \setminus (S_e \cup \{(u, w)\}) \cup \{(w_{3i-1}, w_{3i}) \mid i = 1, 2, \dots, 2a_e - 1\}$ . Notice that the new  $S$  is still an IM of size no smaller than the old  $S$ . Repeating this for all edges, we obtain that  $|S| \geq 2\sum_e a_e - m + \alpha$  and  $|S_e| \geq 2a_e - 1$ . Let  $I$  be the set of all nodes  $v$  such that  $(v, v') \in S$ . Clearly,  $|I| = |S \setminus \bigcup_e S_e| = |S| - \sum_e |S_e| \geq |S| - \sum_e (2a_e - 1) \geq \alpha$ . Moreover,  $I$  is an independent set in  $G$ , since for any edge  $e = (u, v)$  in  $G$ ,  $(u, u'), (v, v') \in S$  implies  $|S_e| \leq \lceil \frac{(6a_e-2)-4}{3} \rceil = 2a_e - 2$ .  $\square$

The following is an immediate consequence of Lemma 2.

**Corollary 1.** IM on grid graphs is NP-hard.

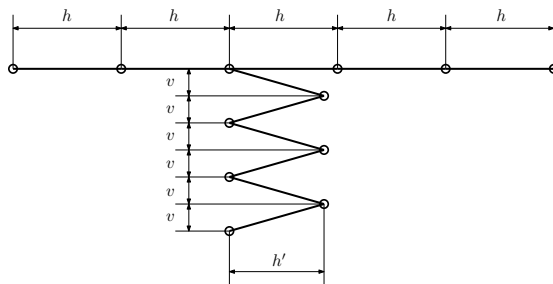
## 2.2 Strip Embedding

We proceed to describe the reduction from IM on grid graphs to our wireless gathering problems. In accordance with the previous section, we denote the input grid graph by  $H_2$ .

The first step is embedding  $H_2$  onto a very narrow horizontal strip. Informally, this is done by scaling down the grid embedding of  $H_2$  in the vertical direction until its vertical extent becomes small enough, while keeping the lengths of all edges just below 1, as they are in the grid embedding. This forces “folding” of vertical paths into a stack of nearly horizontal subpaths with alternating directions (see Figure 2 below for illustration). Since vertical distances generally become very small, some new edges will be introduced to preserve the unit-disk property. Hence the result of this stage, which we denote by  $H_3$ , will be a supergraph of  $H_2$ . The corresponding unit-disk embedding we call the *strip embedding*.

Formally, the construction proceeds as follows. Start from the grid embedding of  $H_2$ , scale it by a factor of  $h = 1 - 2\varepsilon^2$  horizontally, and by a factor of  $v = 2\varepsilon$  vertically, where  $\varepsilon$  is arbitrarily small. Then, process vertical paths one by one. For each vertical path that comes from  $H_1$  (i.e., subdivided edge of  $G$ , rather than  $(v, v')$  edge), move every other vertex in the interior of a vertical path by  $h' = h - 3\varepsilon^2$  to the right. More precisely, if the vertices on the path are  $v_1, v_2, \dots, v_{2t+1}$  (recall that any vertical has even length), then move  $v_2, v_4, \dots, v_{2t}$ . For every vertical edge  $(v, v')$ , move the degree-1 vertex by  $h'$  to the right. Define  $H_3$  to be the unit-disk graph on the resulting set of points.

Figure 2 below (not drawn to scale) illustrates what the strip embedding looks like in the vicinity of a node of degree 3. Any two vertices in the same “column” are connected (those edges are omitted from the drawing for clarity).



**Fig. 2.** Illustration of the strip embedding.

Strip embedding fits into the bounding box  $[0, X] \times [0, Y]$ , where  $X = O(n)$ ,  $Y = O(n\varepsilon)$ , and  $X$  is an integer. Notice that for a vertex  $v$  of  $H_2$  there is a unique integer  $x_v \in [0, X]$  such that the  $x$ -coordinate of  $v$  is either  $x_v h$  or  $x_v h - 3\varepsilon^2$ . Parity of  $x_v$  induces a cut in  $H_3$ .

The key property is that the bipartite subgraph of  $H_3$  consisting of edges that cross this cut is exactly  $H_2$ . Note that this claim is the primary reason for introducing parameters  $s$  and  $t$  in the definition of grid graphs (Section 2.1). By choosing  $s$  and  $t$  to be large enough constants, we make sure that things are well enough separated in the grid embedding, so that in the strip embedding no other

edges are introduced except than those connecting nodes in the same ‘‘column’’ (with the same  $x_v$ ).

### 2.3 Ring Embedding

Now we map several copies of  $H_3$  into a narrow ring (annulus) to obtain a new graph  $H_4$ . Let  $k = 2 \sum_e s_e - m + \alpha$ , where  $\alpha$  is the independence number of  $G$ . Let  $c = z(k + 1) + k - 2$  for an large enough integer  $z$  and let  $R = \frac{ch}{\pi}$  be the inner radius of the ring. Create  $c$  copies of each vertex from the strip embedding of  $H_3$  by mapping it via the following  $c$  functions

$$(x, y) \rightarrow f_i(x, y) = (r_i(y) \cos \phi_i(x), r_i(y) \sin \phi_i(x)) \quad i = 0, 1, \dots, c - 1,$$

where

$$r_i(y) = R + 2(i \bmod (2X))Y + y \quad \phi_i(x) = \frac{2ih + x}{R}.$$

Graph  $H_4$  is defined to be the unit-disk graph on the resulting points.

We prove that each copy has the same connectivity as  $H_3$ . Non-edges in the strip embedding have length at least  $\sqrt{(2h - h')^2 + v^2} \geq \sqrt{1 + 6\varepsilon^2}$ . We analyze how much they are shrunk by  $f_i$ . For  $R \geq \frac{X}{3\varepsilon}$ , we have

$$\begin{aligned} \|f_i(x_1, y_1) - f_i(x_2, y_2)\|^2 &= (y_1 - y_2)^2 + 4r_i(y_1)r_i(y_2) \sin^2 \frac{x_1 - x_2}{2R} \\ &\geq (y_1 - y_2)^2 + 4R^2 \sin^2 \frac{x_1 - x_2}{2R} \\ &\geq (y_1 - y_2)^2 + (1 - 3\varepsilon^2)(x_1 - x_2)^2 \\ &\geq (1 - 3\varepsilon^2)(1 + 6\varepsilon^2) > 1, \end{aligned}$$

so all non-edges remain non-edges. Edges in the strip embedding have length at most  $h = 1 - 2\varepsilon^2$ . We analyze how much they are stretched by  $f_i$ . For  $R \geq \frac{2XY}{\varepsilon^2}$ , we have

$$\begin{aligned} \|f_i(x_1, y_1) - f_i(x_2, y_2)\|^2 &= (y_1 - y_2)^2 + 4r_i(y_1)r_i(y_2) \sin^2 \frac{x_1 - x_2}{2R} \\ &\leq (y_1 - y_2)^2 + 4(R + 2(i \bmod (2X))Y + Y)^2 \left( \frac{x_1 - x_2}{2R} \right)^2 \\ &\leq (y_1 - y_2)^2 + \left(1 + \frac{4XY}{R}\right)^2 (x_1 - x_2)^2 \\ &\leq \left(1 + \frac{4XY}{R}\right)^2 (1 - 2\varepsilon^2)^2 \leq (1 + 2\varepsilon^2)^2 (1 - 2\varepsilon^2)^2 < 1, \end{aligned}$$

so all edges remain edges. Previous calculations also show that it suffices to compute  $f_i$  up to additive error of  $\varepsilon^3$ , which is important for polynomial running time.

Notice that each node  $v$  has coordinates  $(r_v \cos \frac{\phi_v}{R}, r_v \sin \frac{\phi_v}{R})$ , where  $\phi_v$  is either  $x_v h$  or  $x_v h - 3\varepsilon^2$  for some integer  $x_v$  between 0 and  $2c - 1$ . As in the previous section, parity of  $x_v$  induces a cut in  $H_4$ . It is not hard to check that

vertices  $u$  and  $v$  are connected in  $H_4$  if and only if (i)  $x_u = x_v$  or (ii) they belong to the same copy of  $H_3$  and are connected in  $H_3$ . From this we conclude that the subgraph of  $H_4$  consisting of edges that cross the cut is isomorphic to  $c$  disjoint copies of  $H_2$ . Hence its maximum IM is of size  $ck$ . For any fixed  $i \in [0, 2c - 1]$ , define  $V_i$  to be the set of vertices of  $H_4$  with  $x_v = i$ . Due to rotational symmetry of the construction, there is an IM in  $H_4$  such that each  $V_i$  is incident to equally many (i.e.,  $k$ ) matching edges. Such a matching is obtained by applying some fixed optimal solution for  $H_3$  to each copy.

Furthermore, we claim that in any feasible solution, the number of edges adjacent to any  $V_i$  cannot be much more than  $k$ . Consider the feasible solution that maximizes the number of edges adjacent to any  $V_i$ . Consider the  $V_i$  adjacent to most edges, say  $k'$ . By maximality, the solution contains at least  $\frac{1}{4}$  of all edges adjacent to this  $V_i$  that are vertical in the grid embedding (in fact, closer to  $\frac{1}{3}$ , but  $\frac{1}{4}$  suffices for the analysis). Replicate the solution adjacent to this  $V_i$  for all  $V_j$  where  $j$  has the same parity as  $i$ . The new solution has size  $ck'$  and contains at least  $\frac{1}{4}$  of *all* vertical edges. However, the new solution may be infeasible; to restore feasibility, it suffices to remove all horizontal edges. By property (iv) of Lemma 1, horizontal edges comprise at most a  $\frac{\delta}{1+\delta}$ -fraction of all edges. Hence, after removing all horizontal edges, the size of the solution is at least  $ck'(1 - \frac{4\delta}{1+\delta})$ . This implies  $k' \leq k(1 - \frac{4\delta}{1+\delta})^{-1}$ . Since  $\delta$  can be arbitrarily small in Lemma 1, we can abuse notation and from now on denote  $(1 - \frac{4\delta}{1+\delta})^{-1}$  by  $1 + \delta$ .

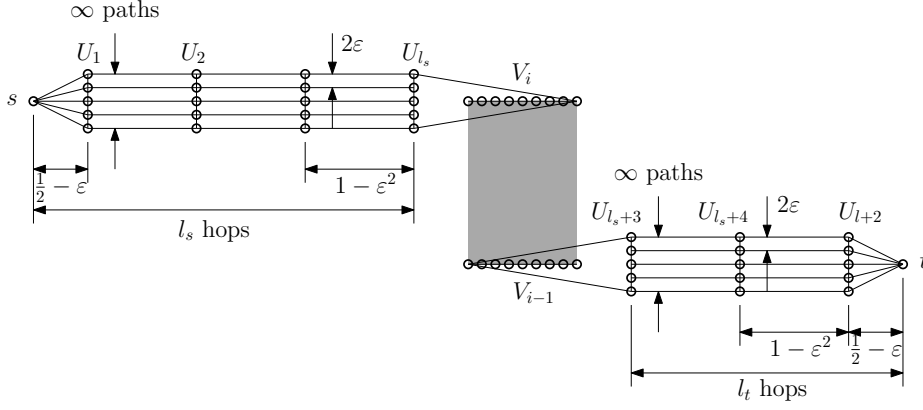
## 2.4 Connecting Paths

We place node  $s$  at the center of the ring, and add nodes in order to connect it to each  $V_i$  with  $i$  odd. For each  $i$ , nodes are placed so that they induce a collection of paths that run parallel and very close to each other (see Figure 3 below for illustration). The number of paths in the collection for a given  $i$  should be thought of as very large. That can be achieved if the paths are close enough to each other.

Formally, the following conditions should hold.

- The paths are of the same length, and two nodes on different paths can be adjacent only if they have the same hop-distance from  $s$ . Nodes on the same path are not adjacent unless they are consecutive (i.e., each path is equal to the subgraph induced by its vertices).
- The first node on each path (closest to  $s$ ), is at most  $\frac{1}{2}$  away from  $s$ . Therefore, the first nodes on all paths form a clique.
- The last node on each path (farthest from  $s$ ) is connected to all nodes of  $V_i$  for one fixed  $i$ , and no other nodes.

It is clear that these conditions can be satisfied for small enough  $\varepsilon$  (depending on  $n$ ), and large enough  $R$  (depending on  $n, \varepsilon$ ). We skip the formal details of the construction to keep the presentation simple; most of the geometry can be inferred from Figure 3.



**Fig. 3.** Illustration of the construction of parallel connecting paths.

Next we place another node  $t$  somewhere outside the ring, and connect it with parallel paths to all  $V_i$  with  $i$  even. Figure 3 shows how to do this for one such  $V_i$ , but it is clear that the paths for all  $V_i$  cannot be “straight” like that one. Fortunately, the above construction allows for two connectivity-preserving modifications: (i) turning the path by a small (polynomial in  $\varepsilon$ ) angle in each step, and (ii) keeping the same direction while modifying the length of the step by a small constant fraction of  $\varepsilon^2$ . Armed with these, we construct curved paths as illustrated in Figure 4. In the curved (solid) parts one exploits modification (i), and in the straight (dashed) parts one exploits modification (ii) to make the lengths of all paths equal.

Let  $l_s$  and  $l_t$  be the length of a parallel path to  $s$  and  $t$ , respectively. Define  $l = l_s + l_t$ . Since  $\varepsilon$  is independent of the size of the input ( $H_2$ , that is),  $l, l_s, l_t$  can be bounded by polynomial functions of the input size.

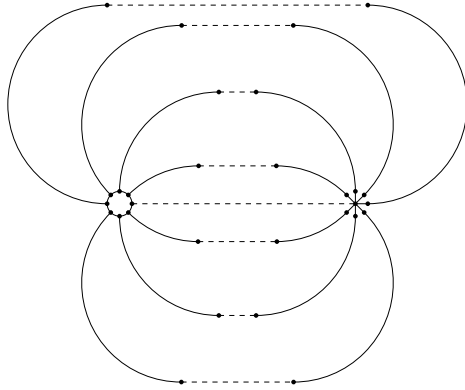
We define  $H_5$  to be the resulting graph. For the next two sections, it is also convenient to define  $U_i$  to be the vertices of  $H_5$  that are exactly  $i$  hops away from  $s$ . In particular,  $U_0 = \{s\}$ , and  $U_{l+3} = \{t\}$ .

## 2.5 NP-Completeness of T-WG

Let  $s$  be the (only) source, and let  $t$  be the sink. Suppose that a batch of  $zck$  packets arrive at  $s$  every  $zck + 1$  time steps. Let there be a total of  $q$  batches.

**Lemma 3.** *If  $H_2$  has an IM of size  $k$ , there is a schedule with maximum latency at most  $2zck + zk + z + k + l - 2$ .*

*Proof.* The schedules for all batches are identical, only shifted in time by  $zck + 1$  steps. We describe the schedule for one batch. All  $zck$  packets go from  $s$  to  $U_1$  in  $zck$  steps, one by one. They travel on parallel paths until they reach  $U_{l_s}$ , and then move to  $U_{l_s+3}$  in smaller groups of size  $ck$  as follows. Each group is sent from  $U_{l_s}$  to  $U_{l_s+1}$  in  $k$  steps (in sub-groups of  $c$ ), then from  $U_{l_s+1}$  to  $U_{l_s+2}$  in



**Fig. 4.** Details of the construction of parallel paths to  $t$ . The small circle on the left represents the ring, and the nodes on it represent sets  $V_i$ . The “central” node on the right is  $t$ . Each geometric curve represents a set of parallel paths connecting one  $V_i$  with  $t$ , closely following the curve (e.g., the horizontal one in the middle is shown in Figure 3). Curved (solid) parts have curvature small enough for modification (i) to be applicable. Horizontal (dashed) parts are long enough.

one step, and then from  $U_{l_s+2}$  to  $U_{l_s+3}$  in  $k$  steps. Note that consecutive groups’ schedules are overlaid, i.e., while one group is being sent from  $V_{l_s+2}$  to  $V_{l_s+3}$ , the next group is being sent from  $V_{l_s}$  to  $V_{l_s+1}$ . That way, all packets move from  $U_{l_s}$  to  $U_{l_s+3}$  in

$$zck \left( \frac{1}{c} + \frac{1}{ck} \right) + k = zk + z + k$$

steps. Finally, they travel on parallel paths to  $t$ , and get delivered to  $t$  one by one. Clearly, maximum latency of one batch is  $2zck + zk + z + k + l - 2$ . It is not hard to check that schedules for different batches do not interfere with each other. Hence the maximum latency for all packets is the same as the maximum latency for a single batch.  $\square$

**Lemma 4.** *If  $H_2$  has no IM of size  $k$ , maximum latency of any schedule is at least  $2zck + z'(k + 1 + \frac{1}{k-1}) + k + l - 3$ , where  $z' = \frac{z}{1+\delta}$ .*

*Proof.* Suppose the maximum IM in  $H_2$  is of size  $k' \leq k - 1$ . Packets can be transferred from  $U_{l_s}$  to  $U_{l_s+2}$  with “rate” at most  $ck'(1 + \delta)$  packets per  $k' + 1$  steps. This is a direct consequence of the fact that in any IM, any column is adjacent to at most  $k'(1 + \delta)$  edges (see the last paragraph of Section 2.3). Therefore, a batch moves from  $U_{l_s}$  to  $U_{l_s+3}$  in at least

$$zck \cdot \frac{k' + 1}{ck'(1 + \delta)} = \frac{kz(k' + 1)}{k'(1 + \delta)} \geq \frac{k^2z}{(k - 1)(1 + \delta)} = z'(k + 1 + \frac{1}{k - 1})$$

steps.

Suppose that there is a batch that departs  $s$  in  $ck$  steps and also gets delivered to  $t$  in  $zck$  steps. Then, we can assume without loss of generality that all packets

from this batch leave  $U_1$ , reach  $U_{l_s}$ , leave  $U_{l_s+3}$  and reach  $U_{l+2}$  together (we can always modify the schedule without increasing the maximum latency to make this true). It follows that its maximum latency is at least

$$2zck + z'(k + 1 + \frac{1}{k-1}) + k + l - 3.$$

Now consider the other case, i.e., suppose that every batch takes at least  $zck+1$  steps to either depart from  $s$  or arrive at  $t$ . Let  $r$  be the number of batches that depart from  $s$  in at least  $zck + 1$  steps. Recall from the beginning of this section that  $q$  is the total number of batches. Then it takes at least  $q(zck + 1) + r$  steps for all packets to depart from  $s$ , and at least  $q(zck + 1) + q - r$  steps for all packets to arrive at  $t$ . Since the hop-distance between  $U_1$  and  $t$ , as well as the distance between  $s$  and  $U_{l+2}$ , is  $l + 2$ , we have that the total length of the schedule for all packets is lower-bounded by both  $q(zck + 1) + q - r + l + 2$  and  $q(zck + 1) + r + l + 2$ . Therefore, it is at most  $q(zck + 1) + q/2 + l + 2$ . The last batch is released in step  $(q - 1)(zck + 1)$ . It follows that the maximum latency is at least  $zck + q/2 + l + 3$ . If we set

$$q/2 \geq zck + z'(k + 1 + \frac{1}{k-1}) + k - 6$$

we get that the length of the schedule is at least  $2zck + z'(k + 1 + \frac{1}{k-1}) + k + l - 3$ , as required.  $\square$

**Theorem 1.** T-WG is NP-complete.

*Proof.* T-WG is obviously in NP. Choose  $\delta$  small enough so that  $z'(k+1+\frac{1}{k-1}) \geq z(k+1+\frac{1}{2(k-1)})$ . The gap between the lower bound of Lemma 4 and the upper bound of Lemma 3 is at least  $\frac{z}{2(k-1)} - 1$ . Choosing  $z > 2(k-1)$ , the gap is positive, that is, our T-WG instance has a schedule of length  $2zck + zk + z + k + l - 2$  if and only if  $H_2$  has an IM of size  $k$ . Together with Corollary 1, this implies that T-WG is NP-hard.  $\square$

## 2.6 NP-Completeness of S-WG

Recall from Section 2.3 that  $c = z(k + 1) + k - 2$  for some large enough integer  $z$ . At time zero,  $p = zck + c$  packets are released at  $U_{l_s}$  so that equally many are adjacent to each  $V_i$ ,  $i = 1, 3, \dots, 2c - 1$ . The sink is  $t$ . Since the release time for all packets is zero, maximum latency is equal to the total length of the schedule.

**Lemma 5.** If  $H_2$  has an IM of size  $k$ , there is a schedule of length  $p + l_t + 3$ .

*Proof.* Suppose that  $H_2$  has an IM of size  $k$ . We describe the schedule of required length. To avoid complicated notation, we keep the presentation informal. We divide the packets into groups  $P$  and  $Q$ , containing  $c$  and  $zck$  packets, respectively.

The schedule for  $P$  is as follows. All packets of  $P$  depart from  $U_{l_s}$  at time 1, and travel from  $U_{l_s}$  to  $U_{l_t+2}$  in  $l_t + 2$  steps using parallel paths. This is possible because  $P$  is small enough that it can move from  $U_{l_s}$  to  $U_{l_s+3}$  in 3 steps, without waiting. Finally,  $P$  gets delivered to  $t$  in  $|P|$  steps.

The schedule for  $Q$  is as follows.  $Q$  departs from  $U_{l_s}$  starting from time 3, but they don't depart all at once. Instead, they move from  $U_{l_s}$  to  $U_{l_s+3}$  in smaller groups of size  $ck$ , as in the proof of Lemma 3. We saw that they need  $zk + z + k$  steps for that. Notice that in this case, due to our choice of  $c$ , this is exactly  $c + 2$  steps. After reaching  $U_{l_s+3}$ ,  $Q$  travels to  $U_{l_t+2}$  on parallel paths, and gets delivered to  $t$  in  $|Q|$  steps.

Now we verify that the schedules of  $P$  and  $Q$  never interfere with each other. This is obvious for all time steps, except perhaps when  $Q$  get near the sink, at which point it may interfere with packets of  $P$  being delivered to  $t$ . We argue that this does not happen.  $Q$  starts departing the sources two steps later than  $P$ . Also,  $P$  needs 3 steps to move from  $U_{l_s}$  to  $U_{l_s+3}$ , while  $Q$  needs  $c + 2$ . Otherwise,  $P$  and  $Q$  proceed in the same fashion on parallel paths. It follows that  $Q$  arrives at  $U_{l_t+2}$  exactly  $c + 1$  steps later than  $P$ ; at that point  $P$  is already delivered to  $t$ .

To finish the proof, we compute the length of the schedule.  $P$  departs  $U_{l_s}$  at time 1, arrives at  $U_{l_t+2}$  at time  $l_t + 2$ , and gets delivered at time  $c + l_t + 2$ . Delivery of  $Q$  starts at time  $c + l_t + 4$  and completes by time  $zck + c + l_t + 3$ .  $\square$

**Lemma 6.** *If  $H_2$  has no IM of size  $k$ , the length of any schedule is at least  $p + l_s + 4$ .*

*Proof.* Suppose the maximum IM in  $H_2$  is of size  $k' \leq k - 1$ . Let  $\tau$  be the length of the schedule. Let  $Q$  be the last batch of packets delivered to  $t$  consecutively (i.e., in  $|Q|$  consecutive steps).

Suppose  $|Q| \geq zck$ . Since packets of  $Q$  are delivered consecutively, they are all at  $U_{l_t+2}$  no later than step  $\tau - |Q|$ . We can assume without loss of generality that they are all at  $U_{l_s+3}$  no later than step  $\tau - |Q| - (l_t - 1)$ , i.e., that they travel together on parallel paths from  $U_{l_s+3}$  to  $U_{l_t+2}$ . Packets can be transferred from  $U_{l_s}$  to  $U_{l_s+2}$  with "rate" at most  $ck'(1 + \delta)$  packets per  $k' + 1$  steps. Therefore,  $Q$  moves from  $U_{l_s}$  to  $U_{l_s+3}$  in at least

$$zck \cdot \frac{k' + 1}{ck'(1 + \delta)} \geq \frac{kz(k' + 1)(1 - \delta)}{k'} \geq \frac{k^2z(1 - \delta)}{k - 1} = (c - k + 2 + \frac{z}{k - 1})(1 - \delta)$$

steps. It follows that

$$\begin{aligned} \tau &\geq zck + l_t - 1 + (c - k + 2 + \frac{z}{k - 1})(1 - \delta) \\ &= (p + l_t + 4) + (c - k + 2 + \frac{z}{k - 1})(1 - \delta) - (c + 5), \end{aligned}$$

which exceeds  $p + l_t + 4$  for large enough  $z$ .

Now suppose  $|Q| < zck$ , and let  $P$  denote all packets outside of  $Q$ . Suppose that  $P$  is delivered in  $|P|$  consecutive steps. Clearly,  $|P| \geq c + 1$ , so there is

some odd  $i$  such that *at least two* packets from  $P$  cross the cut through  $V_i$ . The distance from  $U_{l_s}$  to  $U_{l+2}$  is  $l_t + 2$ , but one of these two packets takes at least  $l_t + 3$  steps to reach  $U_{l+2}$ , because it has to wait one step when moving between  $U_{l_s}$  and  $U_{l_s+3}$ . Since  $|P|$  is delivered consecutively, no packets can be delivered before these two reach  $U_{l+2}$ , hence no packets are delivered in the first  $l_t + 3$  steps. Since packets are delivered in exactly two batches ( $P$  and  $Q$ ), with at least one “empty step” separating them, we have  $\tau \geq (l_t + 3) + (p + 1) = p + l_t + 4$ .

Finally, suppose that  $P$  is not delivered consecutively. Since  $U_{l_s}$  and  $t$  are  $l_t + 3$  hops apart, no packets are delivered in the first  $l_t + 2$  steps. Since packets are delivered in three or more batches, with at least two “empty steps” separating them, we have  $\tau \geq (l_t + 2) + (p + 2) = p + l_t + 4$ .  $\square$

**Theorem 2.** S-WG in NP-complete.

*Proof.* Membership in NP is obvious, while NP-hardness follows directly from Lemma 5, Lemma 6, and Corollary 1.  $\square$

### 3 Discussion and Future Work

This paper initiates the study of wireless gathering problems on unit-disk graphs, by proving that it is NP-complete to decide the existence of a gathering schedule with given maximum latency. This holds even if all packets are originate from the same node *or* if all packets are released at the same time. Our hardness results are much weaker than those for the 3D unit-ball model [3].

There is a wealth of open problems in this area. We believe that our current techniques suffice to establish NP-completeness even when all packets are released *both* at the same node *and* at the same time. We are currently working on this.

An alternative objective function to consider is the *sum of latencies* (which is equivalent to average latency). There are strong inapproximability results for it in the 3D case [3].

One can consider less restricted (and more realistic) models of radio propagation: quasi-unit-disk graphs [4], general disk graphs (in which different nodes are allowed to have different radio ranges) etc.

Finally, there is the question of approximation algorithms, centralized and distributed. For S-WG, a 3-approximation is easy to obtain – simply find any shortest  $s$ - $t$  path and send a packet along that path in every third step. This is exactly the priority greedy algorithm of [3] applied to our setting. The question is whether one can do better. An example from [3] shows that simply forwarding packets along shortest  $s$ - $t$  paths does not work, even in the case of unit-disk graphs in the plane.

As for T-WG, there are strong inapproximability results in the 3D case. Algorithms proposed in [3] perform well under a different metric (resource augmentation). Actually, they do not rely on geometry, so they also work for arbitrary communication/interference graphs. In the 2D case we may be able to do better.

## References

1. Hari Balakrishnan, Christopher L. Barrett, V. S. Anil Kumar, Madhav V. Marathe, and Shripad Thite. The distance-2 matching problem and its relationship to the mac-layer capacity of ad hoc wireless networks. *IEEE Journal on Selected Areas in Communications*, 22:1069–1079, 2004.
2. J.-C. Bermond, N. Morales, S. Perennes, J. Galtier, and R. Klasing. Hardness and approximation of gathering in static radio networks. *Pervasive Computing and Communications Workshops, IEEE International Conference on*, 0:75–79, 2006.
3. Vincenzo Bonifaci, Peter Korteweg, Alberto Marchetti-Spaccamela, and Leen Stougie. Minimizing flow time in the wireless gathering problem. *ACM Transactions on Algorithms*. Accepted for publication.
4. Fabian Kuhn, Roger Wattenhofer, and Aaron Zollinger. Ad Hoc Networks Beyond Unit Disk Graphs. *Wireless Networks*, 14(5):715–729, 2008.
5. V. S. Anil Kumar, Madhav V. Marathe, Srinivasan Parthasarathy, and Aravind Srinivasan. End-to-end packet-scheduling in wireless ad-hoc networks. In *Proceedings of the fifteenth annual ACM-SIAM symposium on Discrete algorithms*, SODA '04, pages 1021–1030, Philadelphia, PA, USA, 2004. Society for Industrial and Applied Mathematics.
6. Y. Shiloach. *Arrangements of Planar Graphs on the Planar Lattice*. PhD thesis, Weizmann Institute of Science, Rehovot, Israel, 1976.
7. Larry J. Stockmeyer and Vijay V. Vazirani. NP-Completeness of Some Generalizations of the Maximum Matching Problem. *Inf. Process. Lett.*, 15(1):14–19, 1982.
8. Leslie G. Valiant. Universality Considerations in VLSI Circuits. *IEEE Trans. Computers*, 30(2):135–140, 1981.