

# Constructing Interference-Minimal Networks

Marc Benkert<sup>1\*</sup>, Joachim Gudmundsson<sup>2</sup>, Herman Haverkort<sup>3\*\*</sup>, and Alexander Wolff<sup>1\*</sup>

<sup>1</sup> Department of Computer Science, Karlsruhe University, P.O. Box 6980, D-76128 Karlsruhe, Germany. <http://i11www.ilkd.uka.de/algo/group>

<sup>2</sup> NICTA<sup>\*\*\*</sup>, Sydney, Australia, [joachim.gudmundsson@nicta.com.au](mailto:joachim.gudmundsson@nicta.com.au)

<sup>3</sup> Department of Computer Science, TU Eindhoven, The Netherlands, [cs.herman@haverkort.net](mailto:cs.herman@haverkort.net)

**Abstract.** A wireless ad-hoc network can be represented as a graph in which the nodes represent wireless devices, and the links represent pairs of nodes that communicate directly by means of radio signals. The *interference* caused by a link between two nodes  $u$  and  $v$  can be defined as the number of other nodes that may be disturbed by the signals exchanged by  $u$  and  $v$ . Given the position of the nodes in the plane, links are to be chosen such that the maximum interference caused by any link is limited and the network fulfills desirable properties such as connectivity, bounded dilation or bounded link diameter. We give efficient algorithms to find the links in two models. In the first model, the signal sent by  $u$  to  $v$  reaches exactly the nodes that are not farther from  $u$  than  $v$  is. In the second model, we assume that the boundary of a signal's reach is not known precisely and that our algorithms should therefore be based on acceptable estimations. The latter model yields faster algorithms.

## 1 Introduction

Wireless ad-hoc networks consist of a number of wireless devices spread across a geographical area. Each device has wireless communication capability, some level of intelligence for signal processing and networking of the data, and a typically limited power source such as a small battery.

This paper studies networks that do not depend on dedicated base stations: in theory, all nodes may communicate directly with each other. In practice however, this is often a bad idea: if nodes that are far from each other would exchange signals directly, their signals may interfere with the communication between other nodes within reach. This may cause errors, so that messages have to be sent again. Communicating directly over large distances would also require sending very strong signals, since the necessary strength depends at least quadratically

---

\* Supported by grant WO 758/4-2 of the German Science Foundation (DFG).

\*\* Part of this research was done while HH was working at Karlsruhe University, supported by the European Commission, FET open project DELIS (IST-001907).

\*\*\* Funded by the Australian Government's Backing Australia's Ability initiative, in part through the Australian Research Council.

on the distance (in practice the dependency tends to be cubic or worse). Both issues could lead to rapid depletion of the devices' limited power sources.

Therefore it is advisable to organize communication between nodes such that direct communication is restricted to pairs of nodes that can reach each other with relatively weak signals that will not disturb many other nodes. We model such a network as a graph  $G = (V, E)$ , in which the vertices  $V$  represent the positions of the mobile devices in the plane, and the links (or: edges)  $E$  represent the pairs of nodes that exchange signals directly. Communication between nodes that do not exchange signals directly should be routed over other nodes on a path through that network. According to Prakash [13], the basic communication between direct neighbours becomes unacceptably problematic if the acknowledgement of a message is not sent on the same link in opposite direction. Hence, we will assume that the links are undirected.<sup>4</sup> Our problem is therefore to find an undirected graph on a given set of nodes in the plane, such that all nodes are connected with each other through the network (preferably over a short path), interference problems are minimized, and direct neighbours in the network can reach each other with signals of bounded transmission radius. In this paper we focus on guaranteeing connectivity and minimizing interference; bounding the transmission radius is an easy extension, which we discuss in the full paper. Since wireless devices tend to move frequently, we need to be able to construct networks with the desired properties fast.

The optimal network structure depends ultimately on the actual communication that takes place. This is generally not known a priori. Therefore we strive to optimize a property of nodes or links in the network that is expected to be a good indicator of the likelihood that interference problems will in fact occur. Assuming that each node can adjust the strength of each signal so that it can just reach the intended receiver, such indicators may be:

**sending-link-based interference of a link  $\{u, v\}$ :** the number of third nodes that are within reach of the signals from the communication over a particular link  $\{u, v\}$  in the network (proposed by Burkhart et al. [2], also studied by Moaveni-Nejad and Li [9])—in other words: the number of nodes that are hindered when the link  $\{u, v\}$  is active. This is the definition of interference we focus on in this paper.

**sending-node-based interference of a node  $u$ :** the number of nodes that receive signals transmitted by  $u$  (proposed by Moaveni-Nejad and Li [9])—in other words: the number of nodes that are hindered when  $u$  is active.

**receiving-node-based interference of a node  $u$ :** the number of nodes transmitting signals that reach  $u$  (proposed by Rickenbach et al. [16])—that is: the number of nodes that may prevent  $u$  from communicating effectively.

*Previous results.* To construct a network that connects all nodes and minimizes the maximum and total sending-link-based interference, we could run Prim's minimum-spanning-tree algorithm [14] with a Fibonacci heap. Assuming that the

---

<sup>4</sup> Nevertheless, most of the algorithms presented in this paper can be extended to work in the directed model, as will be discussed in the full version of the paper.

interference for each feasible link is given in advance, this takes  $\mathcal{O}(m + n \log n) = \mathcal{O}(n^2)$  time, where  $n$  is the number of nodes and  $m$  is the number of eligible links. If all  $\binom{n}{2}$  possible links are considered, we can compute their interference values in  $\mathcal{O}(n^{9/4} \text{polylog } n)$  time (see the proof of Lemma 3). This will then dominate the total running time.

To make sure that nodes are connected by a relatively short path in the network, one could construct a  $t$ -spanner on the given set of nodes. A network  $G$  is said to be a  $t$ -spanner if, for every pair of vertices  $u$  and  $v$ , the length of the shortest path in the network is at most a chosen constant  $t$  times the Euclidean distance between  $u$  and  $v$ . The *dilation* of a graph  $G$  is the smallest  $t$  such that  $G$  is a  $t$ -spanner. Burkhart et al. [2] presented a first algorithm to construct a  $t$ -spanner for given  $t > 1$ . It was later improved by Moaveni-Nejad and Li in [9]. Assuming that the interference for each possible link is again given in advance, the running time of their algorithm is  $\mathcal{O}(n \log n(m + n \log n))$ . If all  $\binom{n}{2}$  possible links are considered, the running time is  $\mathcal{O}(n^3 \log n)$ .

The approach for sending-link-based interference also works for sending-node-based interference, by defining the interference of a link  $(u, v)$  to be the maximum of the sending-node-based interferences of  $u$  and  $v$ . Unfortunately the same approach does not work for receiving-node-based interference. With sending-link-based interference we can decide whether a link causes too much interference independently of the other links that may be active. With receiving-node-based interference this is not possible, so that completely different algorithms would be needed. Rickenbach et al. [16] only give an approximation algorithm for the case where all nodes are on a single line (the *highway model*).

*Our results.* We improve and extend the results of Burkhart et al. and Moaveni-Nejad and Li in two ways.

First, apart from considering networks that are simply connected (spanning trees) and networks with bounded dilation ( $t$ -spanners), we also consider networks with bounded link diameter, that is: networks such that for every pair of nodes  $\{u, v\}$ , there is a path from  $u$  to  $v$  that consists of at most  $d$  links (or: 'hops'), where  $d$  is a parameter given as input to the algorithm. Such  $d$ -hop networks for small  $d$  are useful since much of the delay while sending signals through a network is typically time spent by signals being processed in the nodes rather than time spent by signals actually travelling.

Second, we remove the assumption that the interference of each possible link is given in advance. For each of the three properties (connectivity, bounded dilation or bounded link diameter), we present algorithms that decide whether the graph  $G_k$  with all links of interference at most  $k$  has the desired property. The main idea is that we significantly restrict the set of possible links for which we have to determine the interference, in such a way that we can still decide correctly whether  $G_k$  has the desired property. To find the smallest  $k$  such that there is a network with interference  $k$  and the desired property, we do a combined exponential and binary search that calls the decision algorithm  $\mathcal{O}(\log k)$  times. The resulting algorithms are output-sensitive: their running times depend on  $k$ , the interference of the resulting network.

	exact model (expected)	estimation model (determ.)
spanning tree	$\min\{n^{9/4} \text{polylog } n, nk^2 + n \log n\}$	$n \log n/\varepsilon^2 + n/\varepsilon^3$
$t$ -spanner	$n^2 k \log k + n^2 \log n \log k$	$n^2 \log n \log k/\varepsilon^4$
$d$ -hop network	$\min\{n^{9/4} \text{polylog } n, nk^2\} + n^2 \log n \log k$	$n^2 \log k/\varepsilon^2 + n/\varepsilon^3$

**Table 1.** Running times of our algorithms to find a minimum-interference network with the required property. The running times are given in  $\mathcal{O}$ -notation,  $n$  is the number of nodes,  $k$  is the maximum interference of any link in the resulting network, and  $\varepsilon$  specifies the relative inaccuracy with which a signal’s reach and the dilation of a spanner is known. Worst-case running times for deterministic algorithms for the exact model are slightly worse than the expected running times of the randomized algorithms listed in the table. The listed running times for the estimation model are worst-case.

Our algorithms work for sending-link-based and sending-node-based interference. We present algorithms that optimize these interference measures for two models of the area that is reached by a sender. In the *exact* model, we assume that the signal sent by a node  $u$  to a node  $v$  reaches exactly the nodes that are not farther from  $u$  than  $v$  is. Our algorithms for this model are faster than the algorithm by Moaveni-Nejad and Li [9] for  $k \in o(n)$ . In the *estimation* model, we assume that it is not realistic that the boundary of a signal’s reach is known precisely: for points  $w$  that are slightly farther from  $u$  than  $v$  is, counting  $w$  as being disturbed by the signal sent from  $u$  to  $v$  is as good a guess as not counting  $w$  as disturbed. It turns out that with this model, the number of links for which we actually have to compute the interference can be reduced much further, so that we get faster algorithms, especially for spanning trees with larger values of  $k$ . Our results are listed in Table 1.

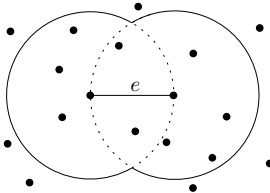
Most of the techniques discussed in this paper generalize to three (or more) dimensions. Details are in the full paper.

This paper is organized as follows. In Section 2 we propose our output-sensitive algorithms for the case of exact interference values, and examine their running times. In Section 3 we introduce our model for reasonable estimations of link interference and describe one algorithm for each of the network properties (connectivity, bounded dilation, and bounded link diameter). All proofs are omitted from this extended abstract but can be found in the full paper.

## 2 Computing exact-interference graphs

We are given a set  $V$  of  $n$  points in the plane in general position. Our aim is to establish a wireless network that minimizes interference. First, we define interference. Let  $u, v$  be any two points in  $V$ . If the edge (link)  $\{u, v\}$  is contained in a communication network, the range of  $u$  must be at least  $|uv|$ . Hence, if  $u$  sends a signal to  $v$  this causes interferences within the closed disk  $D(u, |uv|)$  that has center  $u$  and radius  $|uv|$ . The same holds for  $v$ . This leads to the following definition that was first given by Burkhart et al. [2]. See also Figure 1.

**Definition 1 ([2]).** The sphere of an edge  $e = \{u, v\}$  is defined as  $S(e) := D(u, |uv|) \cup D(v, |uv|)$ . For any edge  $e = \{u, v\}$  in  $V \times V$  we define the interference of  $e$  by  $\text{Int}(e) := |V \cap S(e) \setminus \{u, v\}|$ . The interference  $\text{Int}(G)$  of a graph  $G = (V, E)$  is defined by  $\text{Int}(G) := \max_{e \in E} \text{Int}(e)$ .



**Fig. 1.** The sphere of  $e$ . Here  $\text{Int}(e) = 9$ .

In this section, we will give algorithms to compute interference-minimal networks of three types. The first type is a spanning tree  $\mathcal{T}$ . The algorithm that we use to compute  $\mathcal{T}$  can simply be extended such that the resulting tree  $\mathcal{T}$  not only minimizes  $\text{Int}(\mathcal{T})$  but also  $\sum_{e \in \mathcal{T}} \text{Int}(e)$ . The second type of network is, for an additionally given  $t \geq 1$ , a  $t$ -spanner. And the third type is a  $d$ -hop network, for a given integer  $d > 1$ .

The main idea of the algorithms is the same. For given  $j \geq 0$  let, in the following,  $G_j = (V, E_j)$  denote the graph where  $E_j$  includes all edges  $e$  with  $\text{Int}(e) \leq j$ . Exponential and binary search are used to determine the minimum value of  $k$  for which  $G_k$  has the desired property  $\mathcal{P}$ , see Algorithm 1. We first try  $k = \lfloor \text{upper} \rfloor = 0$ , and compute all edges of  $G_0$ . If  $G_0$  does not have the desired property, we continue with  $\text{upper} = 1$  and then keep doubling  $\text{upper}$  until  $G_{\text{upper}}$  has the desired property. We compute the interference values for each of its edges, and continue with a binary search between  $\text{lower} = \text{upper}/2$  and  $\text{upper}$ . In each step we construct  $G_{\text{middle}}$ , the graph to be tested, by selecting the edges with interference at most  $\text{middle}$  from  $G_{\text{upper}}$ , which had already been computed.

To get a spanning tree, we test with the property  $\mathcal{P} = \text{connectivity}$ . After running Algorithm `MININTERFERENCENETWORK` that finds the minimum  $k$  for which  $G_k$  is connected, we run a standard minimum spanning tree algorithm on  $G_k$ . The result is a tree  $\mathcal{T}$  that minimizes both  $\max_{e \in \mathcal{T}} \text{Int}(e)$  and  $\sum_{e \in \mathcal{T}} \text{Int}(e)$  among all spanning trees. For the  $t$ -spanner and the  $d$ -hop network, the test consists of determining the dilation or the link diameter of the network. We do this with an all-pairs-shortest-paths computation.

Note that the only non-trivial steps in algorithm `MININTERFERENCENETWORK` are the subroutines *FulfillsProperty* and *ComputeEdgeSet*. We first give details on how to implement *ComputeEdgeSet*, that is: how to compute  $E_j$  and the interference values of the edges in  $E_j$  efficiently for any  $j$ .

## 2.1 Computing edge interferences

An edge  $\{u, v\}$  is an *order- $j$  Delaunay edge* if there exists a circle through  $u$  and  $v$  that has at most  $j$  points of  $V$  inside [5].

**Lemma 1.** All edges in  $E_j$  are order- $j$  Delaunay edges.

There is a close connection between order- $j$  Delaunay edges and higher-order Voronoi diagrams that we will use.

---

**Algorithm 1:** MININTERFERENCENETWORK( $V, \mathcal{P}$ )

---

```
// exponential search
upper  $\leftarrow \frac{1}{4}$ , repeat
  lower  $\leftarrow$  upper, upper  $\leftarrow 2 \cdot$  upper
   $E_{\text{upper}} \leftarrow \text{ComputeEdgeSet}(V, \lfloor \text{upper} \rfloor)$ ,  $G_{\text{upper}} \leftarrow (V, E_{\text{upper}})$ 
until FulfillsProperty( $G_{\text{upper}}, \mathcal{P}$ )
// binary search
while upper > lower + 1 do
  middle  $\leftarrow \frac{1}{2}(\text{lower} + \text{upper})$ 
   $G_{\text{middle}} \leftarrow (V, \text{all edges in } E_{\text{upper}} \text{ with interference at most } \text{middle})$ 
  if FulfillsProperty( $G_{\text{middle}}, \mathcal{P}$ ) then upper  $\leftarrow$  middle else lower  $\leftarrow$  middle
return  $G_{\text{upper}}$ 
```

---

**Lemma 2.** (Lemma 2 in [5])

Let  $V$  be a set of  $n$  points in the plane, let  $j \leq n/2 - 2$ , and let  $u, v \in V$ . The edge  $\{u, v\}$  is an order- $j$  Delaunay edge if and only if there are two incident faces,  $F_1$  and  $F_2$ , in the order- $(j+1)$  Voronoi diagram such that  $u$  is among the  $j+1$  points closest to  $F_1$  but not among the  $j+1$  points closest to  $F_2$ , while  $v$  is among the  $j+1$  points closest to  $F_2$  but not among those closest to  $F_1$ .

Since the worst-case complexity of the order- $(j+1)$  Voronoi diagram is  $\mathcal{O}((j+1)(n-j-1))$  [8], it follows that  $\mathcal{O}(nj)$  pairs of points give rise to all order- $j$  Delaunay edges. This is because any two incident faces induce exactly one point pair that corresponds to a Delaunay edge. These pairs can be computed in  $\mathcal{O}(nj2^{c \log^* j} + n \log n)$  expected time [15] (the best-known deterministic algorithm has a worst-case running time that is only slightly worse [4]). Note that this also implies that the number of edges in  $E_j$  is bounded by  $\mathcal{O}(nj)$ .

**Lemma 3.** Given  $n$  points in the plane, (i) any edge set  $E_j$  with  $j \leq n/2 - 2$  can be computed in  $\mathcal{O}(nj^2 + n \log n)$  expected time; (ii) after  $\mathcal{O}(n^{9/4} \text{polylog } n)$  preprocessing time, any edge set  $E_j$  can be computed in  $\mathcal{O}(nj)$  worst-case time.

*Proof.* (i) Computing the order- $(j+1)$  Voronoi diagram and thus all  $\mathcal{O}(nj)$  order- $j$  Delaunay edges takes  $\mathcal{O}(nj2^{c \log^* j} + n \log n)$  expected time. In the full version we explain how to compute the interference of an order- $j$  Delaunay edge, and thus, whether or not it is in  $E_j$ , in  $\mathcal{O}(j)$  time. Thus, computing  $E_j$  takes  $\mathcal{O}(nj^2 + n \log n)$  expected time in total.

(ii) We use results by Matoušek [10] to preprocess the set of points in  $\mathcal{O}(n^{9/4} \text{polylog } n)$  time, so that we can answer range queries with disks and intersections of two disks in  $\mathcal{O}(n^{1/4} \text{polylog } n)$  time. We query this data structure for all  $\mathcal{O}(n^2)$  possible edges, which takes  $\mathcal{O}(n^{9/4} \text{polylog } n)$  time, sort the results for all edges by interference value, and store them. After that, any edge set  $E_j$  can be computed by selecting only those with interference at most  $j$ .  $\square$

## 2.2 The total running time

**Theorem 1.** Algorithm MININTERFERENCENETWORK can run in  $\mathcal{O}(\min\{nk^2 + n \log n, n^{9/4} \text{polylog } n\} + P(n, nk) \log k)$  expected time, where  $n$  is the number of

nodes,  $k$  is the interference of the network output, and  $P(n, m)$  is the running time of *FulfillsProperty* on a graph with  $n$  nodes and  $m$  edges.

*Proof.* During the exponential-search phase, *ComputeEdgeSet* is called  $\mathcal{O}(\log k)$  times to compute an edge set  $E_{\lfloor upper \rfloor}$ , for geometrically increasing values of *upper*. Thus, the last call to *ComputeEdgeSet* dominates and the total expected time spent by *ComputeEdgeSet* is  $\mathcal{O}(nk^2 + n \log n)$  (by Lemma 3). Once the total time spent by *ComputeEdgeSet* has accumulated to  $\Omega(n^{9/4} \text{polylog } n)$ , we compute the interference values for all possible edges at once in  $\mathcal{O}(n^{9/4} \text{polylog } n)$  time, after which we can identify all sets  $E_{\lfloor upper \rfloor}$  easily in  $\mathcal{O}(nk)$  time. In the binary-search phase, selecting edges of  $E_{middle}$  from  $E_{upper}$  takes  $\mathcal{O}(nk)$  time, which is done  $\mathcal{O}(\log k)$  times for a total of  $\mathcal{O}(nk \log k)$ . A total of  $\mathcal{O}(\log k)$  tests for the property  $\mathcal{P}$  on graphs with  $\mathcal{O}(nk)$  edges takes  $\mathcal{O}(P(n, nk) \log k)$  time.  $\square$

For a graph with  $n$  nodes and  $m$  edges, connectivity can be tested in  $\mathcal{O}(n + m)$  worst-case time by breadth-first search. The dilation can be computed in  $\mathcal{O}(nm + n^2 \log n)$  time by computing all pairs' shortest paths, where the length of an edge  $\{u, v\}$  is the Euclidean distance between  $u$  and  $v$ . The link diameter can be computed in the same time (defining the length of every edge to be 1), or in  $\mathcal{O}(n^2 \log n)$  expected time with the all-pairs-shortest-paths algorithm by Moffat and Takaoka [11]. By filling in  $P(n, m)$  in Theorem 1 we get the following:

**Corollary 1.** *We can compute a minimum-interference...*

- (i) ...spanning tree in  $\mathcal{O}(\min\{nk^2 + n \log n, n^{9/4} \text{polylog } n\})$  expected time.
- (ii) ... $t$ -spanner in  $\mathcal{O}(n^2 k \log k + n^2 \log n \log k)$  expected time.
- (iii) ... $d$ -hop network in  $\mathcal{O}(\min\{nk^2, n^{9/4} \text{polylog } n\} + n^2 \log n \log k)$  expected time.

### 3 Estimating interference

In this section we show how to compute interference-minimal networks if nodes  $w$  just outside the intended transmission radius of a node  $u$  may or may not be counted as being disturbed by  $u$ —it may not be realistic anyway to assume that we could predict correctly whether  $w$  will actually be disturbed. We define estimated interference as follows:

**Definition 2.** *Let  $D(u, r)$  be the closed disk centered at  $u$  with radius  $r$ . The  $(1 + \varepsilon)$ -sphere  $S_{1+\varepsilon}(e)$  of an edge  $e = \{u, v\}$  is defined as  $S_{1+\varepsilon}(e) := D(u, (1 + \varepsilon) \cdot |uv|) \cup D(v, (1 + \varepsilon) \cdot |uv|)$ . For  $0 \leq \varepsilon' \leq \varepsilon$  we say that an integer  $I$  is an  $(\varepsilon', \varepsilon)$ -valid estimation of the interference of  $e$  if and only if  $|V \cap S_{1+\varepsilon'}(e) \setminus \{u, v\}| \leq I \leq |V \cap S_{1+\varepsilon}(e) \setminus \{u, v\}|$ .*

We will use  $\varepsilon$ -valid estimation as a shorthand for  $(0, \varepsilon)$ -valid estimation. Our aim is to compute interference-minimal networks based on  $\varepsilon$ -valid estimations of interference. We will do so for a particular assignment  $\text{Int}_\varepsilon : V \times V \rightarrow \mathbb{N}$  of estimations for all edges, which will be explained below. This assignment  $\text{Int}_\varepsilon$  has the following properties that allow for efficient algorithms:

**sparseness** It comes with a set of  $\mathcal{O}(n/\varepsilon^2)$  representative edges  $E'_{n,\varepsilon} \subseteq V \times V$ .  
**representing interference** Each edge  $e \in V \times V$  is represented by an edge  $e' \in E'_{n,\varepsilon}$  s.t.  $\text{Int}_\varepsilon(e) = \text{Int}_\varepsilon(e')$  and  $\text{Int}_\varepsilon(e)$  is an  $\varepsilon$ -valid estimation of  $\text{Int}(e)$ .  
**representing properties** We can test whether the graph  $G_{j,\varepsilon} = (V, E_{j,\varepsilon})$ , with  $E_{j,\varepsilon} = \{e \mid e \in E \wedge \text{Int}_\varepsilon(e) \leq j\}$ , has the desired property  $\mathcal{P}$ , by testing the graph  $G'_{j,\varepsilon} = (V, E'_{n,\varepsilon} \cap E_{j,\varepsilon})$ .

For larger values of  $j$  and  $\varepsilon$ , we have better bounds on the size of  $G'_{j,\varepsilon}$  than for  $G_{j,\varepsilon}$  or  $G_j$  and we get better bounds on the running times of the algorithms to construct and test  $G'_{j,\varepsilon}$  than for the graphs  $G_j$  used in the exact model.

Below we define the edge set  $E'_{n,\varepsilon}$  and the assignment  $\text{Int}_\varepsilon$ . Its sparseness follows from the method of construction. In the full paper we prove that  $\text{Int}_\varepsilon$  correctly represents interference. We describe how to determine the connectivity, dilation, and link diameter of  $G_{j,\varepsilon}$  efficiently by running a test on  $G'_{j,\varepsilon}$ . Our construction uses the well-separated pair decomposition by Callahan and Kosaraju [3], which we briefly review before we define  $E'_{n,\varepsilon}$  and  $\text{Int}_\varepsilon$ .

**Definition 3 ([3]).** Let  $s > 0$  be a real number, and let  $A$  and  $B$  be two finite sets of points in  $\mathbb{R}^2$ . We say that  $A$  and  $B$  are well-separated with respect to  $s$ , if there are two disjoint disks  $D_A$  and  $D_B$  of same radius  $r$ , such that (i)  $D_A$  contains  $A$ , (ii)  $D_B$  contains  $B$ , and (iii) the minimum distance between  $D_A$  and  $D_B$  is at least  $s \cdot r$ .

**Definition 4 ([3]).** Let  $V$  be a set of  $n$  points in  $\mathbb{R}^2$ , and let  $s > 0$  be a real number. A well-separated pair decomposition (WSPD) for  $V$  with respect to  $s$  is a sequence of pairs of non-empty subsets of  $V$ ,  $\{A_1, B_1\}, \{A_2, B_2\}, \dots, \{A_m, B_m\}$ , such that (i)  $A_i$  and  $B_i$  are well-separated w.r.t.  $s$ , for  $i = 1, \dots, m$ , and for any two distinct points  $u$  and  $v$  of  $V$ , there is exactly one pair  $\{A_i, B_i\}$  in the sequence, such that (a)  $u \in A_i$  and  $v \in B_i$ , or (b)  $v \in A_i$  and  $u \in B_i$ . The integer  $m$  is called the size of the WSPD.

Callahan and Kosaraju [3] showed that a WSPD of size  $\mathcal{O}(s^2n)$  can be computed in  $\mathcal{O}(s^2n + n \log n)$  time.

We now define the assignment  $\text{Int}_\varepsilon$  and the set  $E'_{n,\varepsilon}$ . For a well-separated pair  $\{A_i, B_i\}$ , let  $\mathcal{E}_i$  be the set  $\{\{u, v\} \mid u \in A_i, v \in B_i\}$ . To obtain  $\text{Int}_\varepsilon$  we compute an  $(\frac{1}{3}\varepsilon, \frac{2}{3}\varepsilon)$ -valid interference estimation of one exemplary edge  $e_i \in \mathcal{E}_i$  for each pair  $\{A_i, B_i\}$ . We then assign that interference estimation  $\text{Int}_\varepsilon(e_i)$  to all edges in  $\mathcal{E}_i$ , that is,  $\text{Int}_\varepsilon(e) := \text{Int}_\varepsilon(e_i)$  for all edges  $e \in \mathcal{E}_i$ . The set  $E'_{n,\varepsilon}$  consists of the exemplary edges  $e_1, \dots, e_m$  for all well-separated pairs.

In the full paper we show that if we choose the separation constant of the well-separated pair decomposition to be at least  $4 + 16/\varepsilon$ , the estimated interference  $\text{Int}_\varepsilon(e)$  of any edge  $e$  is indeed an  $\varepsilon$ -valid estimation of  $\text{Int}(e)$ . The general algorithm for finding a minimum-interference network based on estimated interferences is given in Algorithm 2.

### 3.1 The total running time

**Theorem 2.** Algorithm `MINESTIMATEDINTERFERENCENETWORK` can run in  $\mathcal{O}(n/\varepsilon^3 + n \log n/\varepsilon^2 + P(n, n/\varepsilon^2) \log k)$  time, where  $n$  is the number of nodes,  $k$  is

---

**Algorithm 2:** MINESTIMATEDINTERFERENCENETWORK( $V, \mathcal{P}$ )

---

```
 $E'_{n,\varepsilon} \leftarrow \emptyset, s \leftarrow 4 + 16/\varepsilon$   
 $W \leftarrow \text{ComputeWSPD}(V, s)$   
 $B \leftarrow \text{ComputeBBDTree}(V)$   
for each  $\{A_i, B_i\} \in W$  do  
  choose an arbitrary edge  $e \in \mathcal{E}_i$   
  query  $B$  to determine a  $(\frac{1}{3}\varepsilon, \frac{2}{3}\varepsilon)$ -valid estimation  $\text{Int}_\varepsilon(e)$  of  $\text{Int}(e)$   
  add  $e$  to  $E'_{n,\varepsilon}$   
// exponential search  
 $upper \leftarrow \frac{1}{4}$   
repeat  
   $lower \leftarrow upper, upper \leftarrow 2 \cdot upper$   
   $E'_{upper,\varepsilon} \leftarrow$  all edges in  $E'_{n,\varepsilon}$  with estimated interference at most  $upper$   
   $G'_{upper,\varepsilon} \leftarrow (V, E'_{upper,\varepsilon})$   
until  $\text{FulfillsProperty}(G'_{upper,\varepsilon}, \mathcal{P})$   
// binary search  
while  $upper > lower + 1$  do  
   $middle \leftarrow \frac{1}{2}(lower + upper)$   
   $E'_{middle,\varepsilon} \leftarrow$  all edges in  $E'_{n,\varepsilon}$  with interference at most  $middle$   
   $G'_{middle,\varepsilon} \leftarrow (V, E'_{middle,\varepsilon})$   
  if  $\text{FulfillsProperty}(G'_{middle,\varepsilon}, \mathcal{P})$  then  $upper \leftarrow middle$  else  $lower \leftarrow middle$   
return  $G'_{upper,\varepsilon}$ 
```

---

the maximum  $\varepsilon$ -valid estimated interference of any edge in the network output, and  $P(n, m)$  is the running time of  $\text{FulfillsProperty}$  on a graph  $G'_{j,\varepsilon}$  with  $n$  nodes and  $m$  edges.

*Proof.* We first construct a well-separated pair decomposition in  $\mathcal{O}(n/\varepsilon^2 + n \log n)$  time, and choose, for each of its  $\mathcal{O}(n/\varepsilon^2)$  pairs, a representative edge. We construct a BBD-tree [1] on the points in  $\mathcal{O}(n \log n)$  time, and do a range query in the BBD-tree for each representative edge to determine its estimated interference in  $\mathcal{O}(1/\varepsilon + \log n)$  time per edge (following the analysis by Haverkort et al. [7]). In total this amounts to  $\mathcal{O}(n/\varepsilon^3 + n \log n/\varepsilon^2)$  time. We then do exponential and binary search in  $\mathcal{O}(\log k)$  steps, each of which takes  $\mathcal{O}(n/\varepsilon^2)$  time to select edges from  $E'_{n,\varepsilon}$  and  $\mathcal{O}(P(n, n/\varepsilon^2))$  time to test the graph, for a total of  $\mathcal{O}(n \log k/\varepsilon^2 + P(n, n/\varepsilon^2) \log k)$  time.  $\square$

In the full paper we prove that we can test if  $G_{j,\varepsilon}$  is connected by testing if  $G'_{j,\varepsilon}$  is connected in time  $\mathcal{O}(n + m)$ . Furthermore, we prove that any minimum spanning tree of  $G'_{j,\varepsilon}$  is a minimum spanning tree of  $G_{j,\varepsilon}$ .

When the dilation of  $G_{j,\varepsilon}$  is  $t$ , an approximate dilation  $t'$  such that  $t/\sqrt{1 + \varepsilon} \leq t' \leq t\sqrt{1 + \varepsilon}$  can be computed in  $\mathcal{O}(n^2 \log n/\varepsilon^2 + mn/\varepsilon^2)$  time with the algorithm by Narasimhan and Smid [12].

The link diameter of  $G_{j,\varepsilon}$  can be computed in  $\mathcal{O}(mn)$  time by doing an implicit breadth-first search from each node. The algorithm is essentially the same as the one in [6] but slightly modified to fit our setting. The search traverses

the edges of  $G'_{j,\varepsilon}$  and the split tree used to construct the well-separated pair decomposition, rather than following the edges of  $G_{j,\varepsilon}$  itself.

By filling in  $P(n, m)$  in Theorem 2 we get the following:

**Corollary 2.** *We can compute a:*

- (i) *minimum-estimated-interference spanning tree in  $\mathcal{O}(n \log n/\varepsilon^2 + n/\varepsilon^3)$  time.*
- (ii)  *$(1 + \varepsilon)t$ -spanner with estimated interference at most  $\min\{k \mid G_{k,\varepsilon} \text{ is a } t\text{-spanner}\}$  in  $\mathcal{O}(n^2 \log n \log k/\varepsilon^4)$  time.*
- (iii) *minimum-estimated-interference  $d$ -hop network in  $\mathcal{O}(n^2 \log k/\varepsilon^2 + n/\varepsilon^3)$  time.*

## References

1. S. ARYA AND D. MOUNT. Approximate Range Searching. *Computational Geometry: Theory & Applications*, 17(3–4): 135–152, 2000.
2. M. BURKHART, P. VON RICKENBACH, R. WATTENHOFER AND A. ZOLLINGER. Does topology control reduce interference? In *Proc. 5th ACM Int. Symp. Mobile Ad Hoc Networking and Computing (MobiHoc)*, pages 9–19, 2004.
3. P. B. CALLAHAN AND S. R. KOSARAJU. A decomposition of multidimensional point sets with applications to  $k$ -nearest-neighbors and  $n$ -body potential fields. *Journal of the ACM*, 42:67–90, 1995.
4. T. M. CHAN. Random sampling, halfspace range reporting, and construction of ( $\leq k$ )-levels in three dimensions. *SIAM J. on Computing*, 30(2): 561–575, 2000.
5. J. GUDMUNDSSON, M. HAMMAR AND M. VAN KREVELD. Higher order Delaunay triangulations. *Computational Geometry: Theory & Appl.*, 23(1):85–98, 2002.
6. J. GUDMUNDSSON, G. NARASIMHAN AND M. SMID. Distance-preserving approximations of polygonal paths. *FST & TCS*, pages 217–228, 2003.
7. H. J. HAVERKORT, M. DE BERG AND J. GUDMUNDSSON. Box-trees for collision checking in industrial installations. *Computational Geometry: Theory & Applications*, 28(2-3):113–135, 2004.
8. D. T. LEE. On  $k$ -nearest neighbor Voronoi diagrams in the plane. *IEEE Transactions on Computing*, 31:478–487, 1982.
9. K. MOAVENI-NEJAD AND X.-Y. LI. Low-interference topology control for wireless ad hoc networks. *Ad Hoc & Sensor Wireless Networks*. Accepted for publ., 2004.
10. J. MATOUŠEK. Range searching with efficient hierarchical cuttings. *Discrete Computational Geometry* 10:157–182, 1993.
11. A. MOFFAT AND T. TAKAOKA. An all pairs shortest path algorithm with expected time  $\mathcal{O}(n^2 \log n)$ . *SIAM Journal on Computing*, 16(6):1023–1031, 1987.
12. G. NARASIMHAN AND M. SMID. Approximating the stretch factor of Euclidean graphs. *SIAM Journal of Computing*, 30(3):978–989, 2000.
13. R. PRAKASH. Unidirectional links prove costly in wireless ad-hoc networks. In *Proc. 3rd Int. Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications (DIAL-M)*, pages 15–22, 1999.
14. R. C. PRIM. Shortest connection networks and some generalisations. *Bell Systems Technical Journal*. p. 1389–1410, 1957.
15. E. A. RAMOS. On range reporting, ray shooting and  $k$ -level construction. In *Proc. 15th Symp. on Computational Geometry*, pages 390–399, 1999.
16. P. VON RICKENBACH, S. SCHMID, R. WATTENHOFER, AND A. ZOLLINGER. A Robust Interference Model for Wireless Ad-Hoc Networks. In *Proc. 5th Int. Worksh. Alg. for Wireless, Mobile, Ad Hoc and Sensor Networks (WMAN)*, CD ROM, 2005.