Constructing Interference-Minimal Networks*

Marc Benkert Joachim Gudmundsson† Herman Haverkort‡ Alexander Wolff§

Abstract

We consider the problem of producing interference-minimal graphs with additional properties such as connectivity, bounded stretch factor or bounded link diameter. We compute exact interference-minimal graphs and estimated interference-minimal graphs. The latter can be computed faster.

1 Introduction

A wireless communication network is commonly modelled by a graph $G = (V, E)$ that consists of a set of points V in the plane and a set of edges E. The nodes represent hosts, while edges represent communication links between nodes. We will assume that the links are undirected. This is a reasonable assumption as a one-way communication becomes unacceptably problematic. It should be noted however, that the algorithms presented in Section 3 can be extended to work for the directed model.

Burkhart et al. [2] propose a definition of interference in a wireless network, and describe two polynomial-time algorithms to construct an optimal spanning tree and a t-spanner in terms of this definition. Their running times are $\mathcal{O}(n^2 \log n)$ for spanning trees and $\mathcal{O}(n^4)$ for t-spanners, however they assume that all interferences are known in advance. We improve and extend these results. Particularly, our algorithms are output-sensitive.

We investigate three different graph classes and show how to compute interference-minimal graphs in each class. First we consider the problem of computing an interference-minimal connected graph, i.e. a spanning tree. The expected running time of our algorithm is $\mathcal{O}(nk(\log^2 k + \log n))$, where k is the minimum interference for which there is a spanning tree.

In addition to connectivity it is often desired that the resulting topology is a t-spanner for some given constant $t > 1$. Given two vertices u and v of a graph G, we will denote by $d_G(u, v)$ the length of the shortest path between u and v in G, and we use $|uv|$ to denote the Euclidean distance between u and v. A network G is said to be a t-spanner if for every pair of vertices u and v it holds that $d_G(u, v) \leq t \cdot |uv|$. We show that an interference-minimal t-spanner can be computed in $\mathcal{O}(n \log k_t({k_t}^2 + n \log n))$ expected time, where k_t is the minimum interference for which there exists a t-spanner.

The third graph class that we consider is the class of d-hop networks. A graph G is said to be a d-hop network if for every pair of vertices u and v, there is a path using at most d links between u and v. The expected running time of our algorithm for the d-hop network is $\mathcal{O}(n \log k_d({k_d}^2 + n \log n))$, where k_d is the minimum interference for which there exists a d-hop network.

It will turn out that our results are only faster than a method that uses *circular range searching* if $k = \mathcal{O}(n^{5/8})$. For dense graphs we argue in Section 4 that it is not realistic to assume exact data. Hosts on the perimeter of the transmission radius may or may not experience interference. Therefore we propose a model based on realistically accurate data, that is, we treat the sphere of an edge as a fuzzy object. This means that hosts lying close to the boundary of the transmission radius either may or may not be counted. Using this cost model we propose algorithms for the tree and the spanner with running times that beat the circular range searching method for all values of k and k_t. However, the running times depend on how fine the approximation should be.

2 Definitions

We start with the definition of interference. For a point $p \in \mathbb{R}^2$ and a real $r \geq 0$ let $D(p, r)$ be the closed disk that has center p and radius r. Given a communication network $G = (V, E)$ with $V \subset \mathbb{R}^2$, for an edge $\{u, v\} \in G$ and u, the range of u must be at least $|uv|$. Thus, all points in $D(u, |uv|)$ experience interferences. As we consider undirected edges, the following definition follows:

Definition 1 ([2]) *The* sphere *of an edge* $e = \{u, v\}$ *is* $S(e) := D(u, |uv|) \cup D(v, |uv|)$. *We define the cost function* Cov *(coverage) on the edge set* $\binom{V}{2}$ *by* $\mathrm{Cov}(e) := \left|V \cap S(e) \setminus \{u, v\}\right|$.

*This work was supported by grant WO 758/4-1 of the German Science Foundation (DFG) and the European Commission within FET Open Projects DELIS.

†Department of Mathematics and Computer Science, TU Eindhoven, h.j.gudmundsson@tue.nl

‡Department of Computer Science, University of Århus, herman@haverkort.net

§Fakultät für Informatik, Universität Karlsruhe, i11www.ira.uka.de/~awolff,~mbenkert

We define the interference $\mathrm{Int}(G)$ *of a graph* $G = (V, E)$ *by* $\mathrm{Int}(G) := \max_{e \in E} \mathrm{Cov}(e)$.

For given $m \geq 0$ let $G_m = (V, E_m)$ denote the graph where E_m includes all edges e with $\mathrm{Cov}(e) \leq m$.

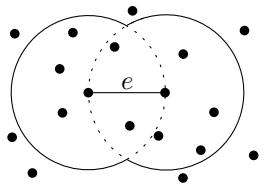

Figure 1: The sphere of e. Here $\mathrm{Cov}(e) = 9$.

3 Computing exact interference graphs

The main idea for computing interference-minimal trees, spanners and d-hop networks is the same. By combining exponential and binary search we determine the minimum graph that has the desired property $\mathcal{P}$, see Algorithm 1.

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

```
E_0 = ComputeEdgeSet(V, 0), G_0 = (V, E_0)
if FulfillsProperty(G_0, P) then
  return G_0
m = 1/2
repeat
  m = 2 · m   {exponential search}
  E_m = ComputeEdgeSet(V, m), G_m = (V, E_m)
until FulfillsProperty(G_m, P)
ℓ = ⌊m/2⌋, r = m
repeat
  m = ℓ + ⌈(r − ℓ)/2⌉   {binary search}
  E_m = ComputeEdgeSet(V, m), G_m = (V, E_m)
  if FulfillsProperty(G_m, P) then
    r = m
  else
    ℓ = m
until r = ℓ + 1
return G_r
```

The only non-trivial steps are the subroutines *ComputeEdgeSet* and *FulfillsProperty*. We first detail how to implement ComputeEdgeSet efficiently. We will use the *order-m Delaunay graph* and the *order-m Voronoi diagram*. An edge $\{u, v\}$ is an *order-m Delaunay edge* if there exists a circle through u and v that has at most m points of V inside.

Lemma 1 *All edges E_m are order-m Delaunay edges.*

Proof. Let e be an edge with $\mathrm{Cov}(e) \leq m$. By definition 1 the sphere $S(e)$ contains at most m points. Then, the disk that has e as diameter contains at most m points as it is contained in $S(e)$. □

We use the duality between Voronoi diagrams and Delaunay edges. We need precisely that $\{u, v\}$ is an order-m Delaunay edge if and only if there are two incident faces F_1 and F_2 in the order-$(m+1)$ Voronoi diagram such that u is in the set of points that determine F_1 and v is in the set of points that determine F_2. Details can be found in [4]. We need:

Theorem 2 ([4]) *For $m \leq n/2-2$, the order-$(m+1)$ Voronoi diagram can be computed in $\mathcal{O}(nm \log m + n \log n)$ expected time. There are $\mathcal{O}(nm)$ order-m Delaunay edges.*

Hence, by Lemma 1 we obtain the edge set E_m by testing all $\mathcal{O}(nm)$ order-m Delaunay edges. To do this we first compute in $\mathcal{O}(nm \log m + n \log n)$ total time for each point $p \in V$ the $m + 1$ nearest neighbors of p. Then, one edge test requires $O(m)$ time: we have to check how many of the $m + 1$ nearest neighbors of u lie in $D(u, |uv|)$ and in $D(u, |uv|) \cap D(v, |uv|)$ and analogously for v. With these numbers we can decide whether $\mathrm{Cov}(\{u, v\})$ is at most m. Thus, we have:

Theorem 3 *Given n points in the plane, the edge set E_m can be computed in $\mathcal{O}(nm^2 + n \log n)$ expected time for any $m \leq n/2 - 2$.*

Next, we describe how to implement *FulfillsProperty* for each of the three graph classes.

3.1 Spanning trees

To obtain an interference-minimal spanning tree we first run Algorithm 1 with the property connectivity. Let k be the minimum value for which G_k is connected. The exponential and binary search to determine k require $\mathcal{O}(\log k)$ steps in total. As the size of the edge set of each graph G_m is bounded by $\mathcal{O}(nk)$, the answer to *FulfillsProperty*$(G_m, \mathcal{P})$ can by given in $\mathcal{O}(nk)$ time by running a breadth-first search on G_m. Hence, we can find the interference-minimal connected graph G_k in $\mathcal{O}(nk^2 \log k + n \log n \log k)$ expected time. If we only want to find a spanning tree $\mathcal{T}$ that minimizes $\mathrm{Int}(\mathcal{T})$, it is enough to run a breadth-first search on G_k. We can also run Prim's algorithm using the values $\mathrm{Cov}(e)$ as edge weights. This requires $\mathcal{O}(nk + n \log n)$ time.

Theorem 4 *We can find an interference-minimal spanning tree $\mathcal{T}$ that minimizes $\mathrm{Int}(\mathcal{T})$ and $\sum_{e \in \mathcal{T}} \mathrm{Cov}(e)$ in $\mathcal{O}(nk^2 \log k + n \log n \log k)$ expected time, if $k \leq n/2 - 2$.*

3.2 Spanners

Here, *FulfillsProperty*$(G_m, \mathcal{P})$ must decide whether the current graph G_m is a t-spanner. We do this by computing shortest paths between all pairs of vertices

and by checking for each pair $\{u, v\} \in \binom{V}{2}$ whether $d_{G_m}(u, v) \leq t \cdot |uv|$. Only if the answer is yes for each query, G_m is a t-spanner of V.

For the all-pairs-shortest-path computation we use an algorithm with expected running time $\mathcal{O}(n^2 \log n)$, described by Moffat and Takaoka in [5], i.e., *FulfillsProperty*$(G_m, \mathcal{P})$ can be implemented in time $\mathcal{O}(n^2 \log n)$ if $\mathcal{P}$ is the graph property "t-spanner". Combining this result with Theorem 3 gives:

Theorem 5 *Given $t > 1$, we can find an interference-minimal t-spanner in $\mathcal{O}(n \log k_t({k_t}^2 + n \log n))$ expected time, if $k_t \leq n/2 - 2$.*

3.3 d-hop networks

For testing the d-hop property in G_m we set the weights of all edges in E_m to 1 and run again the all-pairs-shortest-path algorithm. If every shortest path has weight at most d, then G_m is a d-hop network. We obtain time bounds analogous to those in Theorem 5.

Theorem 6 *Given an integer $d > 1$, we can find an interference-minimal d-hop network in $\mathcal{O}(n \log k_d({k_d}^2 + n \log n))$ exp. time, if $k_d \leq n/2 - 2$.*

4 Computing estimated interference graphs

We now assume that exact data is not realistic. Hosts on the perimeter of the transmission radius may or may not experience interference. Therefore we treat the sphere of an edge as a fuzzy object. This means that hosts lying very close to the boundary of the transmission radius either may or may not be counted. We define the estimated interference cost model:

Definition 2 *For fixed $\varepsilon > 0$ the* maximum estimated sphere *of an edge $e = \{u, v\}$ is defined as $S_{\max}(e, \varepsilon) := D(u, (1 + \varepsilon) \cdot |uv|) \cup D(v, (1 + \varepsilon) \cdot |uv|)$, the* minimum estimated sphere *of e is defined as $S_{\min}(e, \varepsilon) := D(u, (1 - \varepsilon) \cdot |uv|) \cup D(v, (1 - \varepsilon) \cdot |uv|)$. Following this notation we define $\mathrm{Cov}_{\max}(e, \varepsilon)$ and $\mathrm{Cov}_{\min}(e, \varepsilon)$ correspondingly. We say that $c \in \mathbb{N}$ is an* ε-valid interference estimation *of e if $\mathrm{Cov}_{\min}(e, \varepsilon) \leq c \leq \mathrm{Cov}_{\max}(e, \varepsilon)$.*

We will present algorithms for finding sparse networks with minimum estimated interference. Here, 'minimum' is meant with respect to a fixed assignment $\mathrm{Cov}(\cdot) : \binom{V}{2} \to \mathbb{N}$ of ε-valid estimations for all edges, which we will determine later. Let G_m now denote the graph that contains exactly the edges that have estimated interferences of at most m, w.r.t. $\mathrm{Cov}(\cdot)$. The algorithms follow Algorithm 1, only the subroutines *ComputeEdgeSet* and *FulfillsProperty* are implemented differently. The basic idea to speed them up is to perform all computations on a sparse graph G_m^-

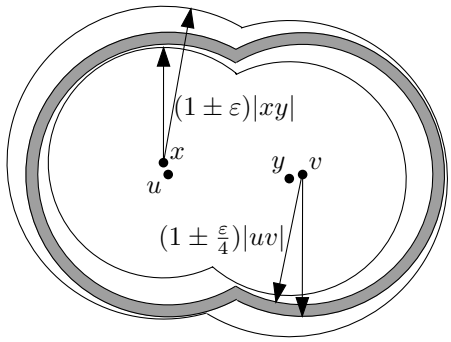

Figure 2: Illustration for Lemma 7.

which closely approximates G_m. We obtain G_m^- using the *well-separated pair decomposition* (WSPD) of Callahan and Kosaraju. G_n^- will contain only one edge for each well-separated pair and G_m^- will simply be the corresponding subgraph of G_n^-. For details about the WSPD see [3]. We briefly recall the basic terminology. Two subsets $A, B \subset V$ are *well-separated* w.r.t. a separation constant s, if there are two closed disks D_A and D_B that have the same radius r and it holds that $A \subset D_A$, $B \subset D_B$ and the distance of D_A and D_B is at least sr. A WSPD of V is a sequence of subsets $\{A_1, B_1\}, \ldots, \{A_m, B_m\}$ such that each pair $\{A_i, B_i\}$ is well-separated and for any two points $u, v \in V$ there is exactly one pair $\{A_j, B_j\}$ with $u \in A_j$ and $v \in B_j$ or $v \in A_j$ and $u \in B_j$. Callahan and Kosaraju [3] showed that a WSPD of size $m = \mathcal{O}(s^2 n)$ can be computed in $\mathcal{O}(s^2 n + n \log n)$ time.

For an well-separated pair $\{A_i, B_i\}$ let E^i denote the set $E^i = \{\{u, v\} \mid u \in A_i, v \in B_i\}$. We obtain G_n^- by computing the WSPD and adding one edge of each E^i. We then compute an $(\varepsilon/4)$-valid interference estimation for the choosen edge of E^i. The next lemma shows that the resulting value is an ε-valid interference estimation for ell edges of E^i. This approach also yields the fixed assignment $\mathrm{Cov}(\cdot)$ of ε-valid interference estimations with respect to which we will compute the estimated interference-minimal graphs.

Lemma 7 *Let $\{A_i, B_i\}$ be a well-separated pair. Let $e_1 = \{u, v\}$ and $e_2 = \{x, y\}$ be two edges of E^i. It holds that an $(\varepsilon/4)$-valid interference estimation for e_1 is an ε-valid interference estimation for e_2, assuming $s \geq (8 + 5\varepsilon)/\varepsilon$.*

If we set the separation constant s to $\mathcal{O}(1/\varepsilon)$, the graph G_n^- has $\mathcal{O}(n/\varepsilon^2)$ edges, and so has G_m^-. The dilation of a graph G is the infimum of the set $\{t \mid G \text{ is a } t\text{-spanner}\}$. We can show that:

Theorem 8 *(i) G_m is connected if and only if G_m^- is connected,*

(ii) if G_m^- has dilation t then G_m has dilation in the range $[t/(1 + \varepsilon), t]$.

Next, we describe how to compute the valid estimations of edge costs efficiently. The idea is to use extended query ranges. For a given ε and a query range Q, the extended query range Q_ε is the set of points lying at L_∞-distance at most εw from Q, where w is the diameter of Q. In our application, the query range will be the union of two disks of radius r, thus we can easily set the values such that Q_ε is the set of points lying at L_∞-distance at most εr from Q.

We will perform ε-approximate counting queries. A simple modification of the BBD-tree by Arya and Mount [1] gives the following theorem:

Theorem 9 *Given a set V of n points in the plane, a constant-complexity query range Q and some $\varepsilon > 0$, ε-approximate range counting queries can be answered in $\mathcal{O}(1/\varepsilon + \log n)$ time using $\mathcal{O}(n \log n)$ preprocessing and $\mathcal{O}(n)$ space.*

Proposition 10 *Given a real value $\varepsilon > 0$ and a constant-complexity range Q, it holds that an ε-approximate range counting query returns an integer N such that $|\{V \cap Q\}| \leq N \leq |\{V \cap Q_\varepsilon\}|$.*

Given an edge e it is straight-forward to see that performing an ε-approximate counting query will give an ε-valid interference estimation for e.

In summary, the graph G_n^- can be computed in $\mathcal{O}(n/\varepsilon^2(1/\varepsilon + \log n))$ deterministic time. Note that we compute G_n^- in a preprocessing step and the graphs G_m^- needed in the exponential and binary search of the algorithms are obtained as subgraphs from G_n^- in $\mathcal{O}(n/\varepsilon^2)$ time. We now show how the sparse graphs G_m^- help to implement the subroutine *FulfillsProperty(G_m, V)* efficiently.

4.1 Spanning trees

According to Theorem 8 (i), we can perform the testing whether G_m is connected by simply testing G_m^-. A breadth-first search in G_m^- requires $\mathcal{O}(n/\varepsilon^2)$ time. Thus, we require $\mathcal{O}(n/\varepsilon^2 \log k)$ time in total for connectivity testing (exponential and binary search), where k is the minimum value for which there is a connected graph. This means that G_k can be computed in $\mathcal{O}(n/\varepsilon^2(1/\varepsilon + \log n))$ time. We then simply run a breadth-first search in G_k^- to obtain an estimated interference-minimal spanning tree.

Theorem 11 *We can find an estimated interference-minimal spanning tree $\mathcal{T}$ in $\mathcal{O}(n/\varepsilon^2(\log n + 1/\varepsilon))$ time.*

By running Prim's algorithm in G_k we could find an interference-minimal spanning tree $\mathcal{T}$ that also minimizes $\sum_{e \in \mathcal{T}} \mathrm{Cov}(e, \varepsilon)$. However, this could require $\Theta(n^2)$ time since G_k can have up to $\Theta(n^2)$ edges.

4.2 Spanners

Let $t^* > 0$ be the given constant for which we want to compute a t^*-spanner. To decide whether G_m has dilation at most t^* one needs to perform a shortest-path query for every pair of points in V. However, since we are only looking for an approximate solution we can use an approximation of t^*. We call t a (c_1, c_2)-*approximate stretch factor* of a graph with dilation t^*, if it holds that $t/c_1 \leq t^* \leq c_2 t$. We apply a result of Narasimhan and Smid [6] that says that a $(1+\varepsilon, 1+\delta)$-approximate stretch factor can be computed in $\mathcal{O}(n\delta^{-2}(\log n + T(n)))$ time for any $\delta > 0$, where $T(n)$ is the time required for a $(1+\varepsilon)$-approximate shortest path query. From Theorem 8 (ii) it follows that $d_{G_m}(u,v) \leq d_{G_m^-}(u,v) \leq (1+\varepsilon) \cdot d_{G_m}(u,v)$ for any two points $u, v \in V$. Thus, performing an exact shortest path query in G_m^- yields a $(1+\varepsilon)$-approximate shortest path for G_m. The running time of such a query is $T(n) = O(n\varepsilon^{-2} \log n)$ if we use Dijkstra's algorithm on the linear-size graph G_m^-. This means computing an approximate stretch factor t of G_m requires $\mathcal{O}(n^2\varepsilon^{-2}\delta^{-2} \log n)$ time in total. The subroutine *FulfillsProperty(G_m, V)* will answer yes if $t/(1+\varepsilon) \leq t^*$. We then know that the output is a graph G_{k_t} for which k_t is at most Int^*, the minimum estimated interference value for any t^*-spanner of V. Setting $\delta = \varepsilon$ and summing up the running times yields the following theorem.

Theorem 12 *Given two real numbers $t^* > 1$ and $\varepsilon > 0$, we can find a $(1+\varepsilon)t^*$-spanner in time $\mathcal{O}(n^2\varepsilon^{-4} \log n \log k_{t^*})$ with estimated interference value at most Int^*.*

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? Proc. 5th ACM International Symp. on Mobile Ad Hoc Networking and Computing (MOBIHOC), 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] J. Gudmundsson, M. Hammar and M. van Kreveld. Higher order Delaunay triangulations. Computational Geometry: Theory & Applications, 23(1):85–98, 2002.

[5] A. Moffat and T. Takaoka An all pairs shortest path algorithm with expected time $\mathrm{O}(n^2 \log n)$. SIAM Journal on Comp., 16(6):1023–1031, 1987.

[6] G. Narasimhan and M. Smid. Approximating the Stretch Factor of Euclidean Graphs. SIAM Journal on Comp., 30(3):978–989, 2000.