UNIVERSITÄT KOBLENZ · LANDAU
Fachbereich 4: Informatik

Reactive Construction of Planar Euclidean Spanners with Constant Node Degree

Bachelorarbeit
zur Erlangung des Grades
BACHELOR OF SCIENCE
im Studiengang Informatik

vorgelegt von

Tim Budweg

Betreuer: M. Sc. Florentin Neumann, Institut für Informatik, Fachbereich Informatik, Universität Koblenz-Landau
Erstgutachter: Prof. Dr. Hannes Frey, Institut für Informatik, Fachbereich Informatik, Universität Koblenz-Landau
Zweitgutachter: M. Sc. Florentin Neumann, Institut für Informatik, Fachbereich Informatik, Universität Koblenz-Landau

Koblenz, im 11 2015

Kurzfassung

Abstract

Insert your abstract in english here. Lorem ipsum dolor sit amet, consectetuer adipi. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipis- cing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lec- tus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

Erklärung

Contents

List of Tables

List of Figures

1 Introduction

Wireless ad-hoc sensor networks are very useful. One can create warning systems for emergency purposes. For instance, deploying many sensor nodes into the sea or a forest to check and caution for tsunamis or fires, respectively.

If a node detects an event and sends a message, it is obvious that this message needs to *arrive* at a certain station. Possibly, this message needs to travel a long distance which one node cannot cover. The solution is to send the message to a neighbor of this node and this node forwards the message to another, and so on, until the message arrives at its destination. While sending from one node to another it may happen that the message gets lost or stuck in a loop, thus, never arriving at its destination. This must be prohibited. There are several routing algorithms which guarantee message delivery, but these require the graph to satisfy a specific property called planarity.

Explaining planarity imagine a graph setup watched from above. It creates a 2d-view of this graph. Planarity says that from this view no two edges are allowed to cross each other except in the endpoints.

One approach to planarize a graph is removing edges. If edges are arbitrarily removed from a graph it may result in a disconnected graph or at least randomly long paths. This needs to be prohibited and can be achieved if a so called *euclidian t-spanner* property is satisfied. With this property satisfied any shortest path in a subgraph H of supergraph P between two points u and v may be not longer than a fixed constant multiplied with the length of this path.

The mechanisms to achieve these properties require the nodes to communicate with each other. A naive approach to gather needed information is to let each node send out beacons. This approach needs a lot of messages and is not robust in terms of network changes. A more sophisticated approach is a reactive one where nodes only send message if they are really needed and otherwise remain silent.

2 Preamble

In this part of this work we define some notations and declare definitions which we will use. In addition, some former mentioned aspects are being formalized.

Nodes which are contained in a graph are denoted in lower case arabic letters and upper case arabic letters are complete graphs which consist of a set of nodes and a set of edges which connect nodes. Let $\bigcirc abc$ be a circle with a, b, c on its border. The circle with center o is denoted with (o). $\triangle abc$ is the triangle with corners a, b and c.

Furthermore, we assume that there are no four points in any graph which are cocircular since the Delaunay Triangulation is in this case not unique anymore. This leads to unnecessary case differentiation.

The Unit Disk Graph of a node set s is denoted as U_s or as U if any node set can be used. This graph contains all nodes of node set s and connects two nodes if and only if their distance between each other is at most 1. In addition, we will make use of the so called *Gabriel Graph*, denoted as GG. It is the graph which contains all nodes of a supergraph U and it contains an edge $uv \in U$ if the Gabriel circle of uv contains no other node. The Gabriel circle of an edge uv is denoted as $disk(u, v)$. It is the circle with u and v on its border and with its center on line uv. In this work U denotes the unit disk graph with unit disk radius $R = 1$.

Another important graph in order to follow this work is the Partial Delaunay Triangulation (PDT) [1]. It is a planar, t-spanner of the Unit Disk Graph (refer to **??**).

The following abbreviations are used throughout this work and introduced here. *Ready To Send (RTS)* and *Clear To Send (CTS)* describe the process of a node pair to interchange messages while starting the desired topology control, namely PDT and RMYS.

2.1 Partial Delaunay Triangulation

The Partial Delaunay Triangulation produces a connected, planar, t-spanner of any connected graph in a reactive approach. No node needs to know its neighborhood. In this part we will see an example of the reactive construction of PDT. First, we define the Partial Delaunay Triangulation as follows:

Definition 2.1. *An edge $uv \in U$ is in $PDT(U_s)$ if either*

(i) $uv \in GG$

(ii) or $\exists w \in U$: *maximizes* $\angle uwv$, $\bigcirc uwv \backslash \{u, v, w\} = \emptyset$ *and* $\sin \angle uwv \geq \frac{|uv|}{R}$, *with $R > 0$ being the unit disk radius.*

The $rPDT$ algorithm taken from chapter 3 of [2] is presented briefly in the following (note that the original PDT definition is from [1]):

Algorithm 1 Partial Delaunay Triangulation

```
Input: any node $u$ of a connected graph $G$
Output: planar, connected view on neighbors of $u$

1: $u$ broadcasts a RTS including its position.
2: All nodes which overhear that message set a timer relative to the Euclidean distance
   to $u$ such that the closest node has the shortest timer.
3: As soon as a timer expires this node sends a CTS including its position.
4: If a node $v$ overhears a CTS from node $t$ it checks whether or not $uv$ violates certain
   geometric conditions which correspond to the conditions in definition 2.1. If it does,
   $v$ cancels its timer and remains silent.
```

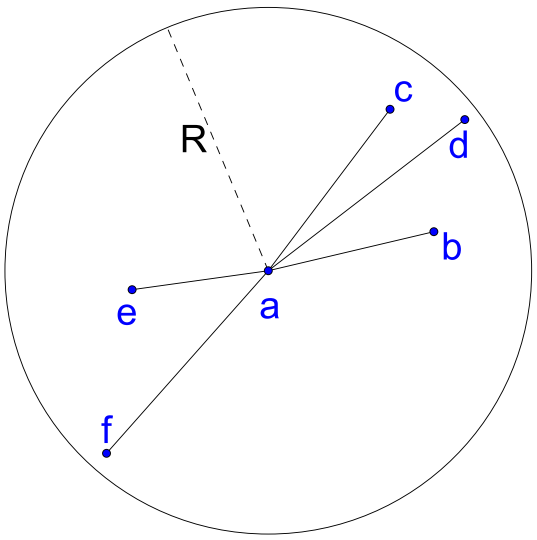

Figure 1: An example graph with unit disk radius R with all edges drawn which are incident on a.

Consider the graph in figure 1. We will use it as an example for the application of rPDT. First, node a sends a RTS and all other nodes set a timer relative to the Euclidean distance to a. Since node e is the closest to a and therefore, there cannot be another node in $disk(a, e)$, its timer fires first and sends a CTS. f overhears that CTS and checks whether or not $e \in disk(a, f)$. Since $e \in disk(a, f)$ is indeed true, $af \notin GG$. However, af is still a PDT edge. The current angle maximizing node e for edge af ($\angle aef$) satisfies the conditions that first, $\bigcirc aef$ is empty and second, that $\sin \angle aef \geq \frac{|af|}{R}$ which means that $\bigcirc aef$ does not overlap the unit disk of

a and hence, a can decide with only one-hop neighborhood information available that $af \in PDT(U)$. These steps are visualized in figure 2. The subsequent steps

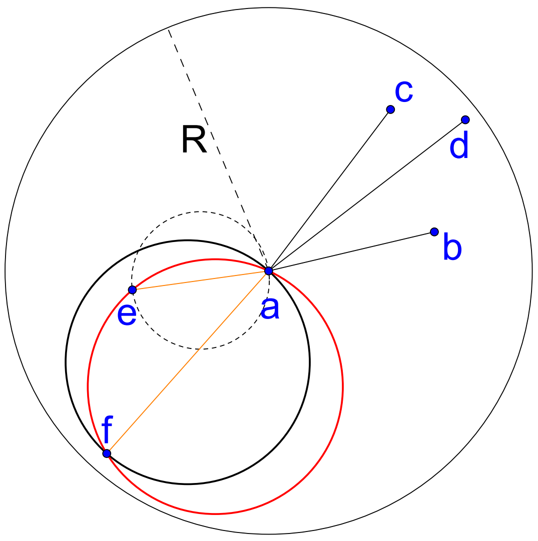

Figure 2: The dashed Gabriel Circle of edge ae and $\bigcirc aef$ proof that both edges ae and af are PDT-edges.

are that b and then c sends a CTS, since both edges are Gabriel edges (refer to figure 3). First, d overhears b' CTS, calculates that $b \in disk(a, d)$ and finally that $\bigcirc abd$ is empty. From that point d did not stop its timer yet. When the CTS from node c arrives at d, $\bigcirc abd$ is not empty anymore and there cannot be another circle which is empty. d cancels its timer since it is no PDT neighbor. Figure 3 shows also the final edge selection of node a after execution of rPDT.

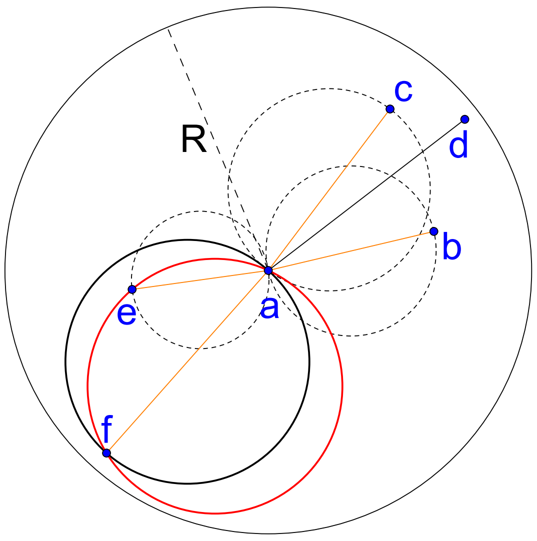

Figure 3: All endpoints of orange edges are PDT neighbors of a. Dotted circles are Gabriel circles and the red circle proofs that af is a PDT edge.

3 Related work

In the past years several topology controls were invented and further developed. We are interested in local algorithms only, and hence, centralized algorithms are ignored in this related work. There are a lot of different approaches with different results. The following is an extract of these approaches and can be divided into two main groups:

1. reactive algorithms
2. algorithms which produce a planar t-spanner with constant node degree

Reactive algorithms generally need less messages as only localized algorithms due to the lack of beaconing. They do not need the whole k-neighbourhood of every node to function, but only a fractional amount of their direct neighbours. As time of writing there are three reactive algorithms:

1. Beaconless Forwarder Planarization (BFP)
2. Guaranteed delivery beaconless forwarding (GDBF) with extension
3. reactive Partial Delaunay Triangulation

First, we describe an algorithm briefly and in the following there is a short section about properties of the produced graph. The BFP-algorithm ([3]) is divided into two phases. First, in the Selection Phase the executing node F starts the algorithm by sending a RTS message. In the following every node, which receives this message, starts a timer corresponding to a specific delay function. The closer a node resides to the executing node, the earlier it answers with a CTS. If a node W overhears a CTS of a node W' it checks whether or not it is contained in a certain area corresponding to node W' and F. This area is defined by geometric regions, in the following denoted as $Reg(A, B)$, with A and B being two nodes specifying this region. The minimum region $Reg(F, W')$ is the Gabriel circle $disk(F, W')$ and the maximum region $Reg(F, W')$ is the Relative Neighbourhood Graph lune over F and W'. The latter describes the area of the intersection of two circles around two neighbouring nodes UV with radii equal to $|UV|$ and with middlepoints U and V, respectively. Different regions cause the algorithm to use different amounts of messages. This will be discussed later.

Suppose W is contained in such an area it cancels its timer and is, henceforth, called a *hidden node*. Hidden nodes further participate in the algorithm. If a hidden node H receives a message from another node T, it memorizes this node if H lies in the former defined region.

The Protest Phase lets hidden nodes protest against violating edges. An edge UV is called a violating edge if there is a node in $Reg(U, V)$. If hidden nodes have nodes they memorize they restart the above timer. As soon as a message from another hidden node W' arrives at hidden node W, the latter checks its memorized nodes: A node X can be removed from the set of memorized nodes if $W' \in Reg(F, X)$. If the timer of a node expires and there are still nodes which are memorized, the node sends a protest message consisting of the violating node. The forwarder node F removes violating edges when it receives protests.

This algorithm performed on each node of a graph G produces a planar subgraph G'. However, G' is not a t-spanner of G and has no constant node degree despite the underlying region (GG, RNG, CNG) (refer to ... for an example of these three regions).

GDBF is a scheme to forward messages in a network. All messages will be greedy forwarded to the node which lies closest to the destination until a node which has no neighbours closer to the destination, called a local minimum, is reached. From that point a recovery mode is used until the local minimum is exited and the algorithm can switch back to greedy mode. In greedy mode the message holder broadcasts a RTS-message to all neighbours. Every neighbour instantiates a timer with length depending on how far the neighbour is away from the destination. Nodes closer to the destination answer earlier. A CTS-message is sent as soon as the timer expires and the message holder forwards the message to its sender. Every other node cancels its timer and remains silent. In recovery mode a RTS message from the message holder is sent as well. Now, all neighbours instantiate a timer corresponding to the distance to the message holder M (closer nodes respond first). If a neighbour N overhears another nodes N' message, it cancels its timer if $N' \in disk(M, N)$. For more detailed information, refer to [4].

GDBF can be extended to reactively produce a planar subgraph of a given input graph. Since this graph is equal to the Gabriel graph, this is not a t-spanner of the input graph and also has no constant node degree.

The understanding of the Partial Delaunay Triangulation is crucial to follow this work and, thus, it is already explained in the preamble. PDT has a constant spanning ratio of at most $\frac{1+\sqrt{5}}{4}\pi^2 \approx 7.98$. In addition, the output is a planar graph, but it has no constant bounded degree.

The second group consists of the following algorithms:

1. H_{PLOS}
2. $\Delta_{11-Spanner}$
3. $PuDel$

H_{PLOS} (Planar Localized Optimum Spanner)[5] produces a planar Euclidean spanner with stretch-factor $1 + \epsilon$ with $\epsilon > 0$ arbitrarily small and constant node degree. However, it needs a node to be aware of its complete 2-hop-neighbourhood.

$\Delta_{11-Spanner}$ [6] constructs a spanner with an upper bound of 7 and a constant node degree of at most 11. The obtained graph is not planar and the algorithm needs a node to know its 4-hop-neighbourhood.

PuDel [7] produces a subgraph which is equal to the subgraph produced by *PDT* [8] and hence, it has an Euclidean stretch-factor of ≈ 7.98, is planar, but has no constant node degree.

4 Algorithm

This chapter introduces the *reactive Modified Yao Step (RMYS)* and explains it's functionality. For the sake of completeness follows a scheme of the Modified Yao Step taken from [9] and how this can be changed to a reactive approach. In addition, there is an explanation of how RMYS operates. Then there is a proof of correctness, followed by an brief analysis of the message complexity and message size of RMYS. Afterwards, we see which properties the graph produced by RMYS obtains and which not.

Algorithm 2 Modified Yao Step

```
Input: planar, connected graph $G$; integer $k \geq 14$
Output: planar, connected graph $G'$ with constant node degree of at most $k$

 1: for each node $p \in G$ do
 2:     Define $k$ disjoint cones of size $2\pi/k$ around $p$.
 3:     Select for each non empty cone the shortest edge.
 4:     for each maximal sequence $s$ of empty cones do
 5:         if $|s| == 1$ then
 6:             Let $nx$ and $ny$ be the incident edges on $p$ clockwise and
 7:             counterclockwise, respectively, from the emtpy cone.
 8:             if either $nx$ or $ny$ has already been selected then
 9:                 select the other edge
10:             else
11:                 Select the shorter edge
12:         else
13:             select the first $\lfloor \frac{|s|}{2} \rfloor$unselected edges incident on $n$ clockwise from $s$
14:             select the first $\lceil \frac{|s|}{2} \rceil$unselected edges incident on $n$ counterclockwise from $s$
    $G'$ is the subgraph of $G$ consisting of all nodes which are in $G$ and all edges which
    fulfil that both endpoints of this edge have selected it.
```

Algorithm 2 does not tell, in particular, how this can be computed on a node. However, my reactive approach, called rMYS, is the following: Since every node knows its PDT-neighborhood (refer to algorithm 3) which is used by the Modified Yao Step, it can execute this algorithm from line 1 to 14 without further knowledge about it's neighborhood and hence, does not need to send any messages at all. My approach of RMYS needs to send a RTS-message to each possible neighbor which send a message back if they did not select this edge. This ensures that only bidirectional edges are used. Since in most cases the edges are bidirectional a node sends only a so called protestmessage if it did not select an edge and thus, protests against the selection of this edge.

The following algorithm is the definition of RMYS. For clarity, notice that both acronyms RMYS and rMYS mean "reactive Modified Yao Step", but former is the algorithm which consists of rPDT, the reactive version of PDT, and rMYS, the reactive way of applying the Modified Yao Step to a planar and connected graph described above.

Algorithm 3 Reactive Modified Yao Step (RMYS)

```
Input: any connected graph $G$; integer $k \geq 14$
Output: planar, connected graph $G'$ with constant node degree of at most $k$

for each node $p \in G$ do
    create the PDT-Neighborhood of $p$ using rPDT
    apply rMYS to $p$ using PDT-graph
    let each neighbor of $p$ create its RMYS-neighbors and send a protest message if $p$
      is not among them causing $p$ to remove this edge
```

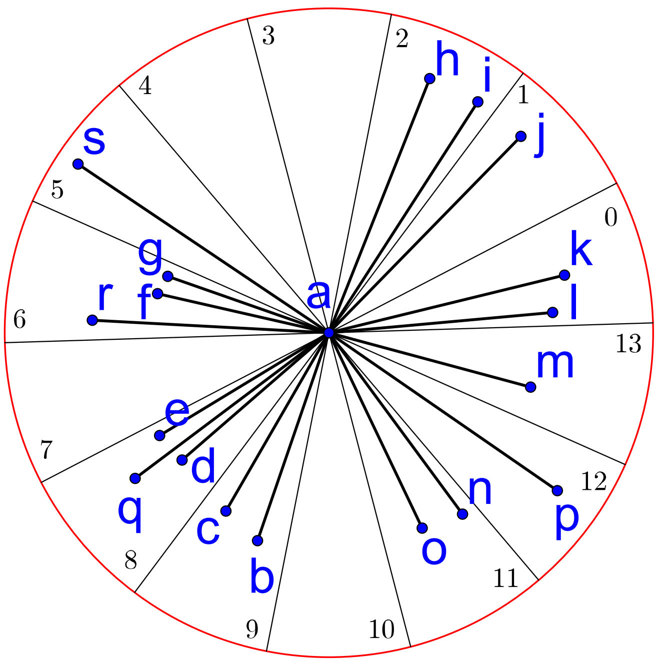

Figure 4: An example graph and all edges drawn which are incident on a. The red circle shows the unit disk of a.

In the following RMYS is presented on the example given by figure 4. Note that this example calculation is only performed on node a. If one may want to create a complete graph, this algorithm must be executed for every node.

First, create the PDT neighborhood of a using rPTD as explained in chapter 2.1. Since nodes p, q, r and s are no PDT neighbors of a and, hence they are not needed anymore, they are omitted in the following figures. Now, the Modified Yao Step starts. Place k equally sized cones around a. It does not matter where these cones start as long as they start at the same place for all nodes. For the purpose of this example assume $k = 14$ and that the first cone starts at a horizontal line and runs counterclockwise. The next step is that a selects for each cone the shortest edge. In figure 5 these steps has been executed. Then, consider all maximal sequences

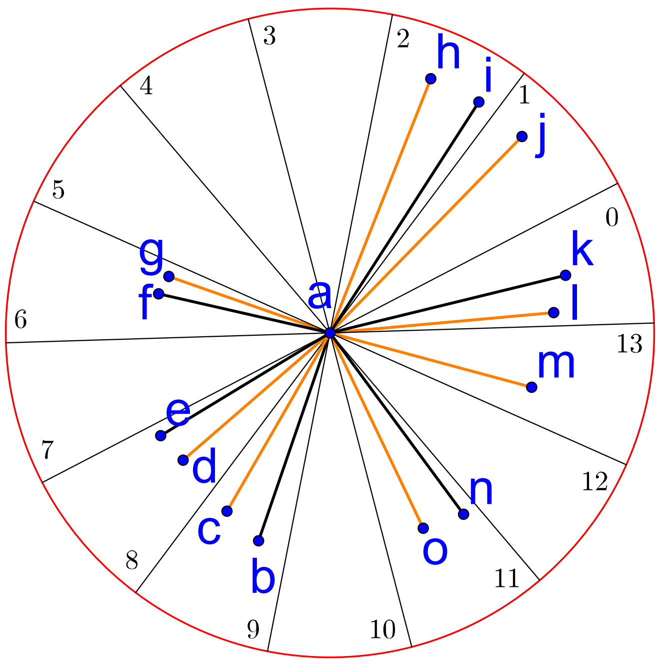

Figure 5: An example graph with equally sized cones around a and all edges drawn which are incident on a. All orange edges has been selected by reason of being the shortest edge in their cone.

of empty cones. Let cones 3 to 5 be the first sequence to be considered. Select the first $\lfloor\frac{|s|}{2}\rfloor = 1$ unselected edges clockwise from s, with s being the length of the sequence. Hence, edge ai is selected now. Counterclockwise the next $\lceil\frac{|s|}{2}\rceil = 2$ unselected edges are selected. For this reason edges af and ae are additionally selected. The next empty sequence to analyze is cone 7. In this case the next edge incident on a clockwise and counterclockwise, respectively, from this cone has to be tested. Since both edges ae and af are selected yet, no edge will be additionally added. Moving on, the next empty sequence which is cone 10 must be inspected. In this example ao is already selected. However, ab is not selected yet. In order to fulfill the algorithm 2 it selects ab at this point in time. The last sequence to analyze is cone 12. For the same reason as in the sequence before *an* gets selected. The last step of the RMYS algorithm is that all of these selected neighbors test whether or not a is also their neighbor. If this is not the case for one node z, this edge is being deleted. In this example every neighbor accepts a as its neighbor. Figure 6 contains the final graph.

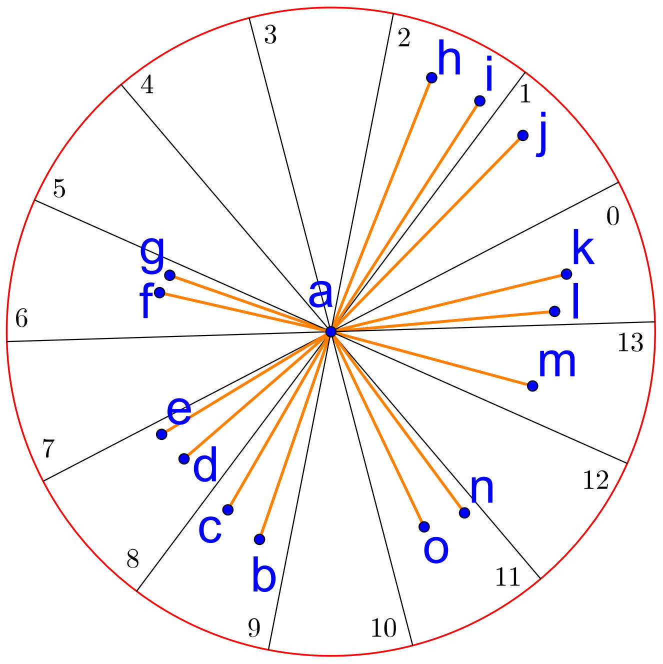

Figure 6: This is the final graph with all edges drawn in orange which are incident on a after the execution of RMYS.

4.1 Proof of correctness

Proof.

$$MYS(PDT) \leftrightarrow RMYS$$

$$MYS(PDT(v)) \overset{a)}{\leftrightarrow} rMYS(rPDT(v))$$

$$MYS(PDT(v)) \overset{b)}{\leftrightarrow} rMYS(PDT(v))$$

$$MYS(PDT(v)) \overset{c)}{\leftrightarrow} MYS(PDT(v))$$

We need to proof that the proposed reactive version of this algorithm is equal to a simple concatenation of first, the Partial Delaunay Triangulation and second, the Modified Yao Step on any node $v \in G$. a) is the fragmentation of the proposition applied to a node v. It is well known that $rPDT$ produces the same graph as the simple local approach, so b) holds true. $rMYS$ does the same calculation as MYS right before the broadcast in the end. Therefore, we need only to look at this broadcast. The executing node v sends a broadcast which must be overheard by all PDT -Neighbors of v. Because of the assumptions that every message arrives and arrives instantaneously, the message cannot get lost. Every informed node checks whether or not it selects v to be in its RMYS-Neighborhood. If yes, it remains silent and otherwise it sends a protest message causing v to remove this edge. Hence, v can check whether or not each node in it's neighborhood accepts this edge. This leads to the same behavior MYS does and therefore, c) is true and completing this proof. ■

4.2 Message Complexity

Let $N_{PDT}(u)$ be the message complexity of PDT creating the neighborhood of Node $u \in G$. First, $rPDT$ needs at most n messages to create the PDT-neighborhood. Next, the executing node sends at most k messages to its neighbors to ask whether they accept their connection. At most k answers come back and therefore $k * 2$. Every one of this k neighbors needs to calculate it's PDT-neighborhood and hence, $k * N_{PDT}(u)$. The following equation put these reflections into one formula.

$$\begin{aligned} N_{RMYS}(u) &= \underbrace{N_{PDT}(u)}_{\theta(n)} + k * \underbrace{2}_{\theta(1)} + k * \underbrace{N_{PDT}(v)}_{\theta(n)} \\ \theta(N_{RMYS}(u)) &= \theta(n) \end{aligned}$$

Since k is a constant it can be omitted in O-Notation. The sum of the same complexity remains in the same complexity and hence, the message complexity of this algorithm is $\theta(n)$.

4.3 Message Size

If the assumption that every node has a unique position holds, this position can be used to identify each node uniquely. Hence, two floats can be used to save this position resulting in a constant number of bits.

4.4 Properties of the RMYS-graph

This section is devoted to the graph-properties the RMYS algorithms inherits. First, it is important to know whether RMYS produces from any connected Unit Disk Graph a connected subgraph. The first part of RMYS, the Partial Delaunay Triangulation, creates from a connected graph a connected subgraph. Since rMYS removes edges it may be possible that the graph will be disconnected. To analyze this we need to recognize that rMYS finds at least one edge per non empty cone. Since these cones around a node p cover the whole area around p and if there is a node in it, there will be an edge for that cone. Hence, the graph cannot become disconnected.

Following this, there is planarity. The reactive approach of the Partial Delaunay Triangulation produces a planar graph and since the rMYS step of the RMYS-algorithm does not add any edges, the planarity property cannot be violated. Hence, RMYS produces a planar graph.

Every node has a constant node degree of at most k. First, for each cone the shortest edge is selected. Resulting in k edges if all cones are not empty. For all other cases let l be the total number of empty cones. Then the first step selects $k - l$ edges in all non empty cones and at most l edges are added in the second step. This leads to a node degree of at most k.

Bibliography

[1] Xiang Yang Li, Ivan Stojmenovic, and Yu Wang. Partial delaunay triangulation and degree limited localized bluetooth scatternet formation. *IEEE Transactions on Parallel and Distributed Systems*, 15(4):350–361, April 2004.

[2] Markus Benter, Florentin Neumann, and Hannes Frey. Reactive Planar Spanner Construction in Wireless Ad Hoc and Sensor Networks. In *In Proceedings of 32nd IEEE International Conference on Computer Communications (INFOCOM)*, pages 2193–2201, Torino, Italy, April 2013.

[3] Stefan Rührup, H. Kalosha, Amiya Nayak, and Ivan Stojmenović. Message-Efficient Beaconless Georouting With Guaranteed Delivery in Wireless Sensor, Ad Hoc, and Actuator Networks. *IEEE/ACM Transactions on Networking*, 18(1):95–108, February 2010.

[4] Mohit Chawla, Nishith Goel, Kalai Kalaichelvan, Amiya Nayak, and Ivan Stojmenović. Beaconless Position-Based Routing with Guaranteed Delivery for Wireless Ad hoc and Sensor Networks. *Acta Automatica Sinica*, 32(6):846–855, November 2006.

[5] Mirela Damian and Sriram V. Pemmaraju. Localized Spanners for Ad Hoc Wireless Networks. *Ad Hoc & Sensor Wireless Networks*, 9(3/4):305–328, 2010.

[6] Iyad A. Kanj and Ge Xia. Improved local algorithms for spanner construction. *Theoretical Computer Science*, 453:54–64, September 2012.

[7] Pengfei Xu, Zhigang Chen, Xiaoheng Deng, and Jianping Yu. A Partial Unit Delaunay graph with Planar and Spanner for Ad hoc Wireless Networks. *Advanced Materials Research*, 267:322–327, 2011.

[8] Florentin Neumann and Hannes Frey. On the Spanning Ratio of Partial Delaunay Triangulation. In *9th IEEE International Conference on Mobile Ad hoc and Sensor Systems (MASS)*, pages 434–442, Las Vegas, Nevada, USA, October 2012. Ieee.

[9] Iyad A. Kanj and Ljubomir Perkovic. On geometric spanners of euclidean and unit disk graphs. *CoRR*, 2008.