UNIVERSITÄT KOBLENZ · LANDAU
Fachbereich 4: Informatik

unikorn
Computer Networks Group

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, 12 2015

Kurzfassung

Bisher existieren keine beaconlosen Algorithmen, welche reaktiv einen planaren euklidischen Spanner produzieren und zugleich konstant ausgangsgradbeschränkte Knoten beinhalten. In dieser Bachelorarbeit werde ich untersuchen, ob die Partial Delaunay Triangulation in Verbindung mit dem Modified Yao Step einen solchen Graph erzeugen kann und es einen Algorithmus gibt, der ihn reaktiv erzeugt. Dieser Algorithmus hat im Vergleich zur derzeitigen State-Of-The-Art sowohl beim Topologieaufbau als auch beim späteren Routing über diese Topologie einen verringerten Energieaufwand zur Folge, was längere Knotenlaufzeiten ermöglicht.

Abstract

Currently, there are no beaconless algorithms which construct planar Euclidean spanners in a reactive way while at the same time providing constant node degree. This work analysis the approach whether or not the Partial Delaunay Triangulation in connection with the Modified Yao Step can create a graph with all of these properties satisfied and searches for an algorithm which creates this graph in a reactive way. This algorithm would need less energy than the current state-of-the-art to construct the topology. In addition, to route messages using that topology leads to a lower energy consumption as well. This enables nodes to work longer with the same amount of power.

Erklärung

Hiermit bestätige ich, dass die vorliegende Arbeit von mir selbständig verfasst wurde und ich keine anderen als die angegebenen Hilfsmittel - insbesondere keine im Quellenverzeichnis nicht benannten Internet-Quellen - benutzt habe und die Arbeit von mir vorher nicht in einem anderen Prüfungsverfahren eingereicht wurde. Die eingereichte schriftliche Fassung entspricht der auf dem elektronischen Speichermedium (CD-Rom).

Mit der Einstellung dieser Arbeit in die Bibliothek bin ich einverstanden.	ja ☒	nein ☐
Der Veröffentlichung dieser Arbeit im Internet stimme ich zu.	ja ☒	nein ☐

Koblenz, den 31. Dezember 2015

Contents

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, if the graph satisfies 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.

The mechanisms to achieve these properties require nodes to be aware of other node's position which require communication between the nodes. A naive approach to gather needed information is to let each node send out messages and every node which hears them answer this node. This approach is called beaconing. It requires 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 messages if they are really needed, e.g. they are definitely nodes of the subgraph, and otherwise remain silent.

This work starts now with a preamble where important definitions and notations are clarified. The next chapter compares several algorithms and their properties. It follows a definition of the algorithm Reactive Modified Yao Step (RMYS) and several graph properties it inherits are presented. In the following some simulation results are presented and discussed. At last, a brief conclusion ends this work.

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. A Graph consists 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.

Furthermore, we assume that there are no four points in any graph which are co-circular 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 and only 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 2.1).

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.

Topology controls are algorithms which create a subgraph of a graph with specific properties. Currently known topology controls can be divided into beacon and beaconless approaches. In a topology control which uses beacons information is gathered periodically and possibly unnecessary information is interchanged which leads to a message overhead. In this work a reactive algorithm is proposed and here, this means that the topology control does not use any beacons and no neighborhood information are available before the algorithms execution.

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 the corresponding 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.

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 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.

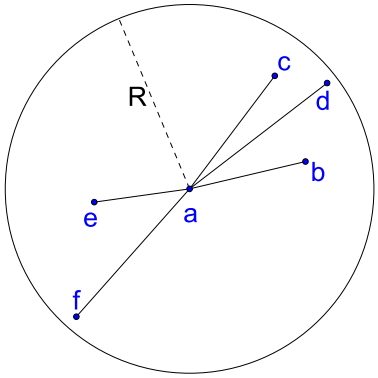

Figure 1: An example graph with unit disk radius R with all edges drawn which are incident on a.

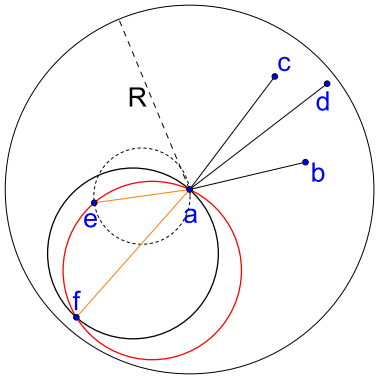

Figure 2: The dashed Gabriel Circle of edge ae and $\bigcirc aef$ proof that both edges ae and af are PDT-edges.

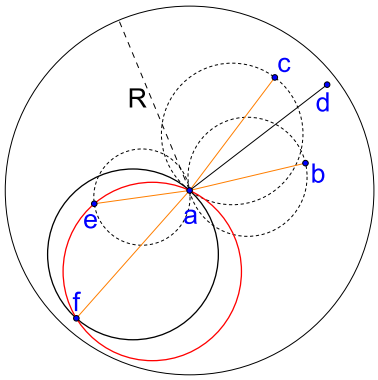

Figure 3: All endpoints of orange edges are PDT neighbors of a. Dotted circles are Gabriel circles and the red circle proofs that af is a PDT edge.

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 $p \in \mathcal{N}$ neighborhood of every node to function, but only a fractional amount of their direct neighbors. 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 (rPDT)

Topology	reactive	planar	Eucl. stretch	Node degree	Ref.
BFP (GG)	✔	✔	$\Theta(\sqrt{n})$	$\mathcal{O}(n)$	[3]
GDBF	✔	✔	$\Theta(\sqrt{n})$	$\mathcal{O}(n)$	[4]
PDT	✔	✔	7.98	$\mathcal{O}(n)$	[1, 5]
H_{PLOS}	✘	✔	$1+\epsilon$	$\mathcal{O}(1)$	[6]
$\Delta_{11-Spanner}$	✘	✘	< 7	11	[7]
$PuDel$	✔	✔	7.98	$\mathcal{O}(n)$	[8]

Table 1: Different topology controls ordered by appearance in this chapter.

First, we describe an algorithm briefly and in the following there is a short section about properties of the produced graph. In addition, table 1 provides an overview of all discussed graphs.

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 Neighborhood Graph lune over f and w'. The latter describes the area of the intersection of two circles around two neighboring 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.

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 graphs Gabriel graph (GG), relative neighborhood graph (RN), circular neighborhood graph (CN) (refer to figure 4 for an example of these three regions).

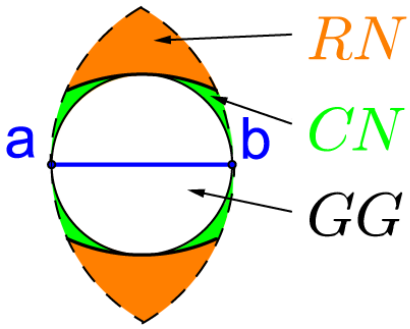

Figure 4: Gabriel circle (GG), relative neighborhood (RN) and circular neighborhood (CN) between a and b.

$GDBF$ 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 neighbors 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 neighbors. Every neighbor instantiates a timer with length depending on how far the neighbor 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 neighbors instantiate a timer corresponding to the distance to the message holder m (closer nodes respond first). If a neighbor 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 $\frac{1+\sqrt{5}}{4}\pi^2 \approx 7.98$ with respect to the Euclidean graph. 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)[6] 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-neighborhood.

$\Delta_{11-Spanner}$ [7] 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-neighborhood.

$PuDel$ [8] is a graph which is equal to the PDT graph [5] and hence, it has an Euclidean stretch-factor of ≈ 7.98 with respect to the Euclidean graph. The graph is planar, it has, however, no constant node degree. Table 1 states that $PuDel$ can be constructed reactively which is true since PDT can be constructed in a reactive way and both graphs are equal.

4 Algorithm

This chapter introduces the *reactive Modified Yao Step (RMYS)* and explains its functionality. For the sake of completeness, it 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 a 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, the 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 its neighborhood and hence, does not need to send any messages at all. This 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

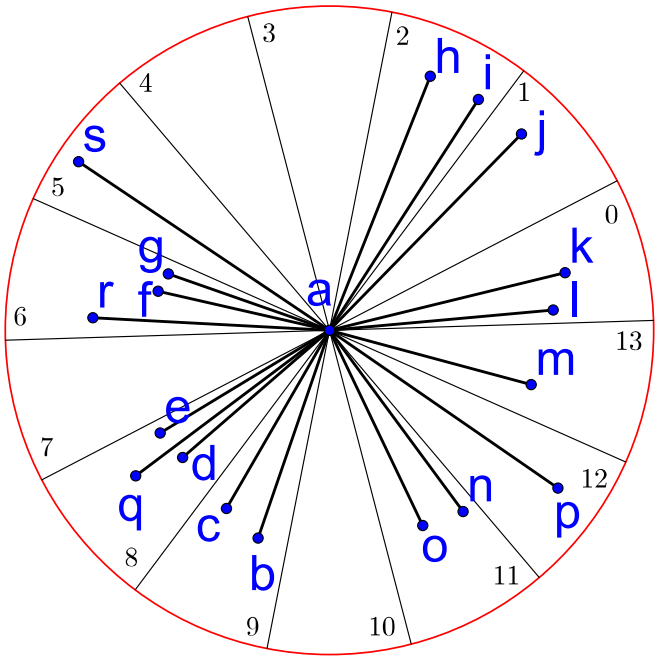

Figure 5: An example graph and all edges drawn which are incident on a. The red circle shows the unit disk of a.

In the following RMYS is presented on the example given by figure 5. 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 6 these steps has been executed. Then, consider all maximal sequences

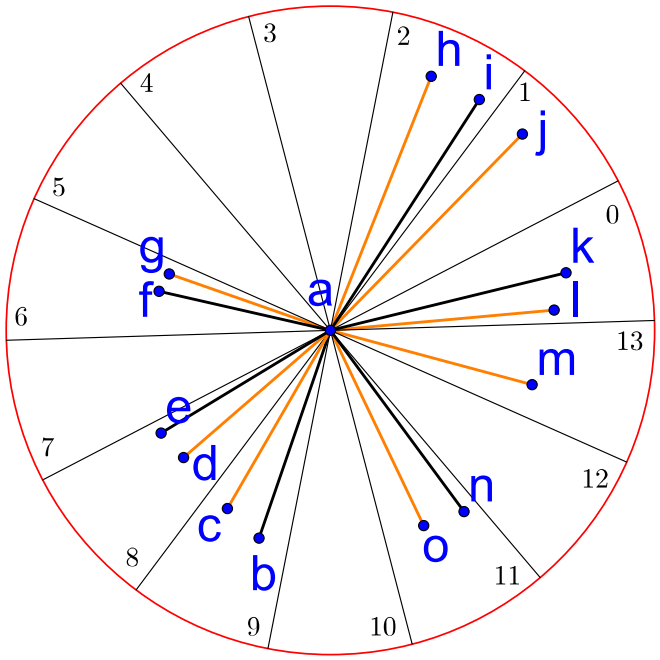

Figure 6: An example graph with equally sized cones around a and all edges drawn which are incident on a. All orange edges have been selected by the reason of being the shortest edge in their cone.

of empty cones. Without loss of generality, 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, none of these edges will be selected again. 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 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, this edge is being unselected. In this example every neighbor accepts a as its neighbor. Figure 7 contains the final graph.

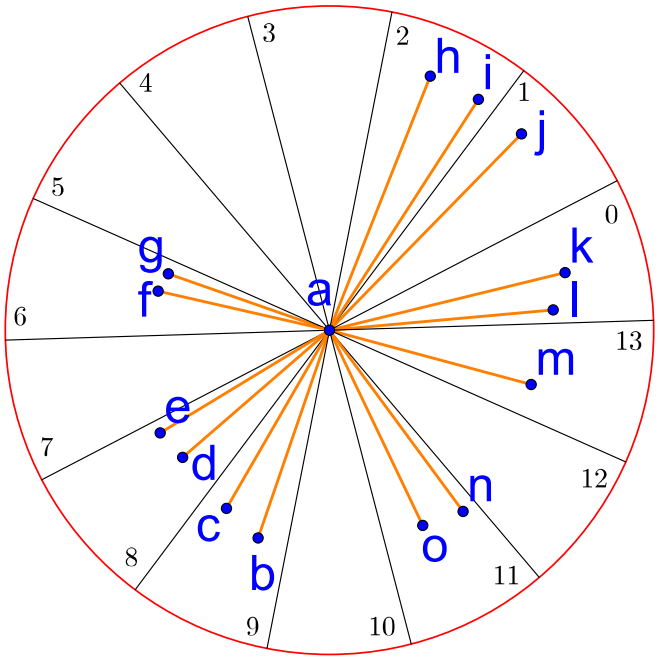

Figure 7: This is the final graph with all edges drawn in orange which are incident on a after the execution of RMYS.

4.1 Proof of correctness

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

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 its 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 one RTS message to its neighbors to ask whether they accept their connection. At most k answers return. Every one of this k neighbors needs to calculate its PDT-neighborhood and hence, $k * N_{PDT}(u)$. The following equation put these reflections into one formula.

$$N_{RMYS}(u) <= \underbrace{N_{PDT}(u)}_{\theta(n)} + \underbrace{1+k}_{\theta(1)} + k * \underbrace{N_{PDT}(v)}_{\theta(n)}$$

$$\mathcal{O}(N_{RMYS}(u)) = \mathcal{O}(n)$$

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 worst case message complexity of this algorithm is $\mathcal{O}(n)$.

4.3 Message Size

If the assumption that every node has an 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.

5 Simulation results

This section is about the simulation results which were achieved. First, these results are presented, then several graph properties RMYS inherits are shown and at last, there is a discussion of the results.

The simulation was performed in Sinalgo[1]. Sinalgo is a simulation framework which can test and validate network algorithms and at the same time it offers a graphical view on the programmed processes as well as a fast batch mode.

There are two different simulation runs. The first one calculated the Euclidean and hop spanning ratio of PDT and RMYS and the second one counted the messages which were needed to construct PDT, RMYS and the amount of messages needed by two hop beaconing. 1000 random connected graphs for each node density from 5 to 20 were used. The numbers of nodes n used for each density were calculated with the following formula:

$$n = round(\frac{D_x \cdot D_y}{\pi \cdot R^2} \cdot density)$$

Hereby, $D_x = 1000$ and $D_y = 1000$ are the dimension of the simulation plane and $R = 100$ is the unit disk radius.

The simulations are based on the following assumptions.

- All nodes are static which means they cannot move.
- All calculations are ordered into synchronous rounds. At the beginning a node receives all messages sent in the round before. It follows a general calculation phase and at last a node can send messages.
- Messages arrive instantaneously and cannot get lost. Every message which was sent, arrives for sure in the round after it was sent.
- There are no 4 nodes which are co-circular.
- All graphs are connected.

Figure 8 shows the measured spanning ratio of RMYS and PDT to the unit disk graph with respect to node density. It is noticeable that both lines seem to be the same line. For density 5 the ratio is approximately 1.30 and density 20 has approximately a spanning ratio of 1.38. In between 5 and 20 the curve is increasing slightly and has no outliers.

Since both lines are almost identical it is possibly true that RMYS also has a constant spanning ratio which is a fractional amount greater than the proven spanning ratio of PDT (refer to [5] for the proven spanning ratio).

[1] http://www.disco.ethz.ch/projects/sinalgo/

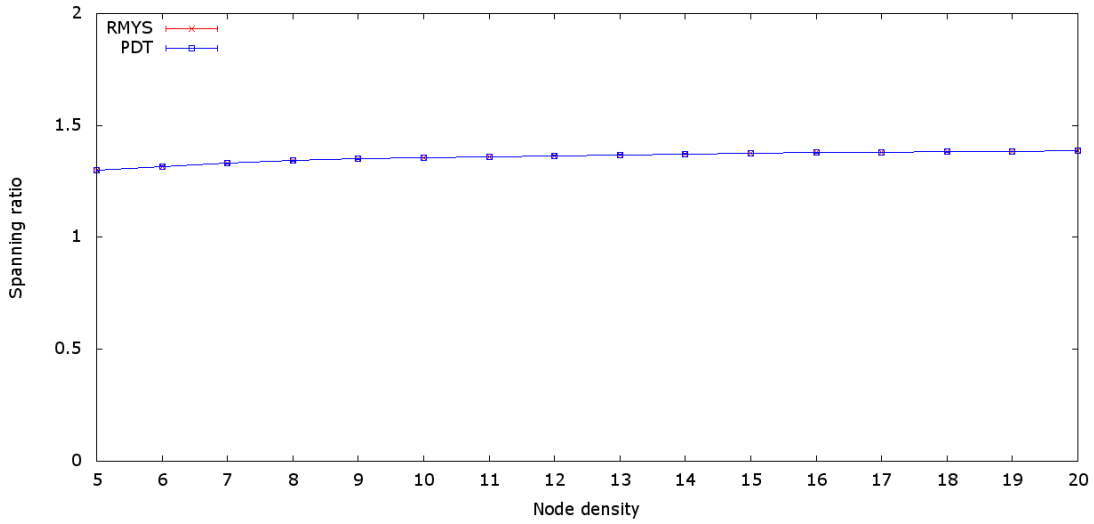

Figure 8: Measured Euclidean spanning ratio of Reactive Modified Yao Step (RMYS) and Partial Delaunay Triangulation (PDT) with respect to the unit disk graph in context to the node density. 1000 Simulations per density.

Figure 9 shows the measured hop spanning ratio of RMYS and PDT to the unit disk graph with respect to node density. The measured values increase from 3 hops at density 5 to approximately 4.6 at density 20.

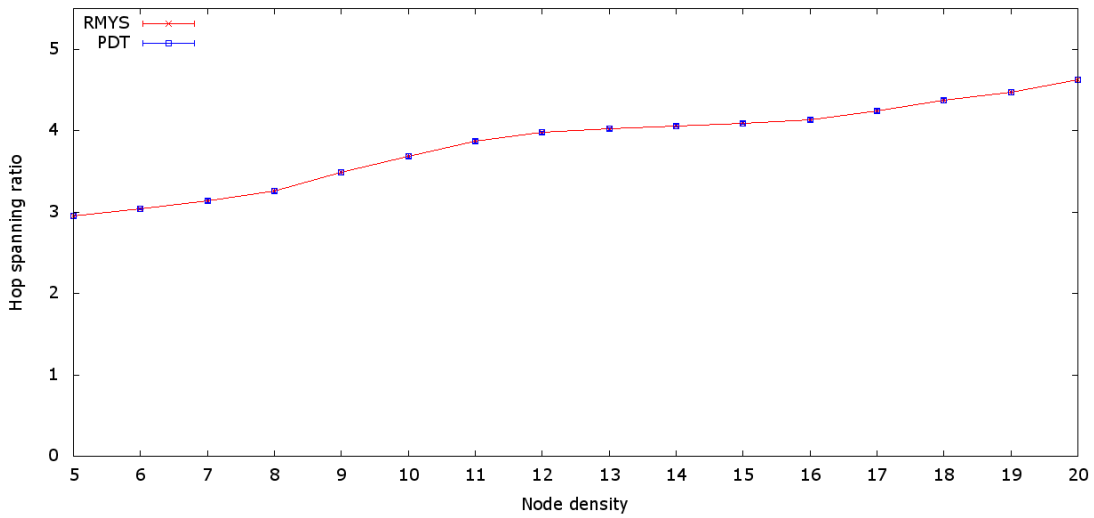

Figure 9: Measured hop spanning ratio of Reactive Modified Yao Step (RMYS) and Partial Delaunay Triangulation (PDT) with respect to the unit disk graph in context to the node density. 1000 Simulations per density.

In figure 10 is the message consumption with respect to node density shown. "PDT on neighbors" shows the amount of messages PDT uses to create the 2 hop PDT neighborhood of one random node within the graph borders and at least unit disk radius $R = 100$ away from the borders to ensure the correct node density. In this example the amount of messages RMYS and PDT on neighbors uses are almost equal. The messages used in 2 hop beaconing are much more. Additionally, the increase from one density to another is greater than the increase in PDT and RMYS. At density 5 beaconing uses almost twice as much messages as RMYS and at density 20 it uses almost tenth times as much messages as RMYS. For reference, the amount of messages which PDT uses to create the 1 hop PDT neighborhood of the same node as above is shown. Notice that PDT is message optimal (refer to [2] for proof) in terms of only nodes which are actually PDT neighbors send messages.

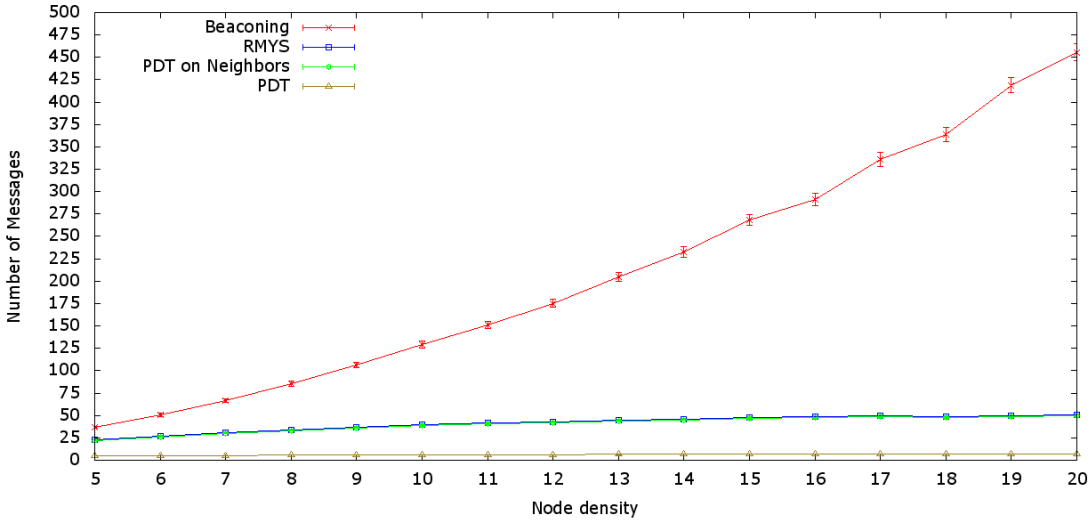

Figure 10: This plot shows the needed messages to construct the RMYS neighborhood on a given node with respect to the node density in a 2-hop beaconing approach (Beaconing) and in a reactive way (RMYS). Additionally, the messages rPDT uses to construct the PDT-neighborhood of a node (PDT) and its neighbors (PDT on Neighbors) are shown. 1000 Simulations per density.

Figure 11 shows the message consumption with respect to node density between RMYS and PDT on neighbors in order to notice the small differences. In fact RMYS uses always one message more than PDT on neighbors. For reference, the message consumption for PDT on one node is shown as well.

Refer to figure 12 to see the average and maximal neighbors of the RMYS and the PDT graph. Density 5 leads to approximately 3.6 neighbors and almost 6 neighbors at density 20. The maximal neighbors differ for both graphs from 6.89

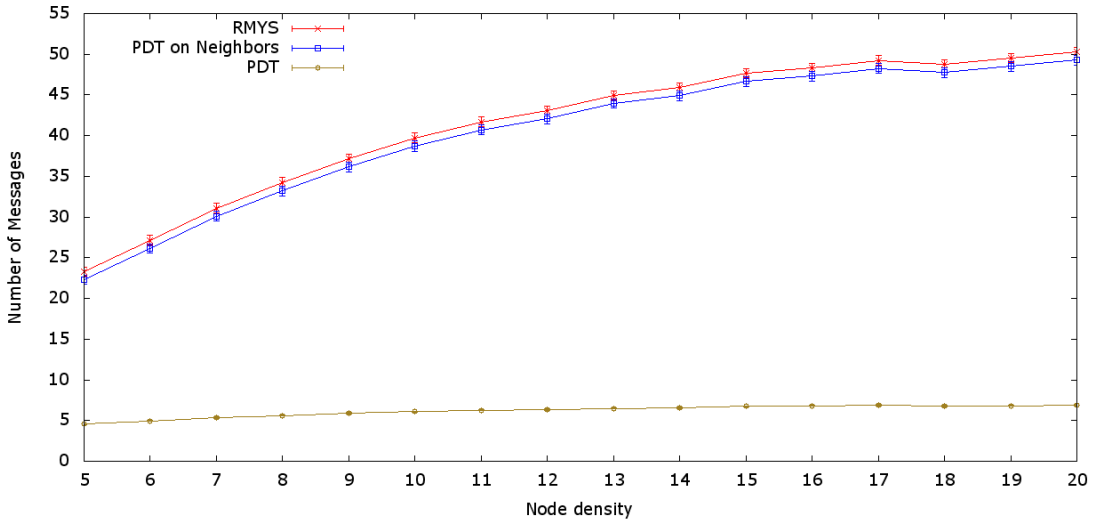

Figure 11: The messages needed to construct the PDT-neighborhood of a node (PDT) and its neighbors (PDT on Neighbors) are shown. Additionally, the message usage of the RMYS neighborhood creation is visualized. 1000 Simulations per density.

at density 5 to approximately 10.35 for RMYS and 10.44 for PDT at density 20. The differences of the maximal neighbors for RMYS and PDT at high densities is explained in chapter 6.

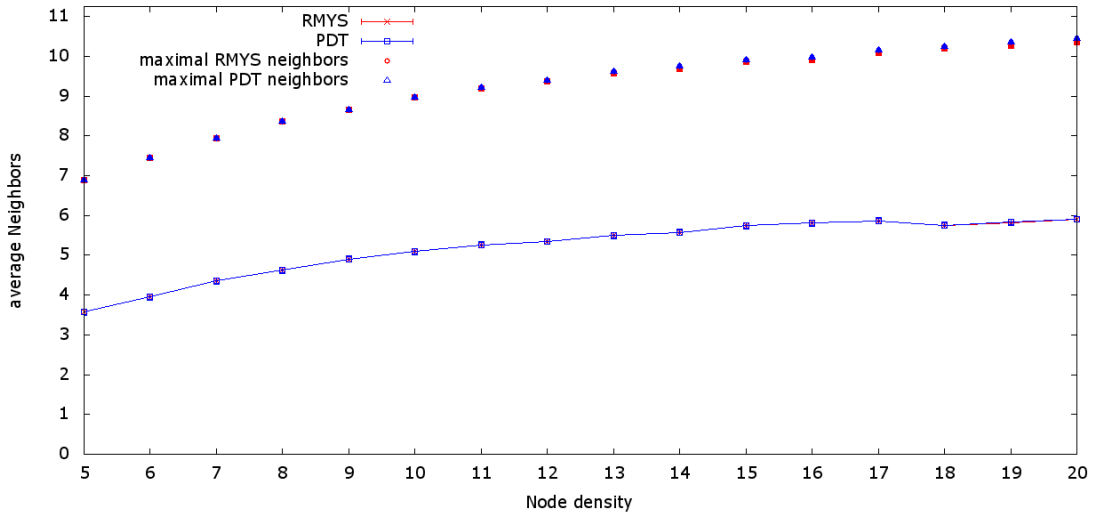

Figure 12: Average and maximal Neighbors of Reactive Modified Yao Step (RMYS) and Partial Delaunay Triangulation (PDT) with respect to node density. 1000 Simulations per density.

Figure 13 shows the amount of neighbors in the subgraphs RMYS and PDT of a random node divided by the amount of neighbors in the unit disk model of the same node. A value of 1 corresponds to all unit disk neighbors are neighbors in the subgraph. At density 5 approximately 0.8 percent of all unit disk neighbors are PDT and RMYS neighbors. At density 20 this percentage dropped to 0.33.

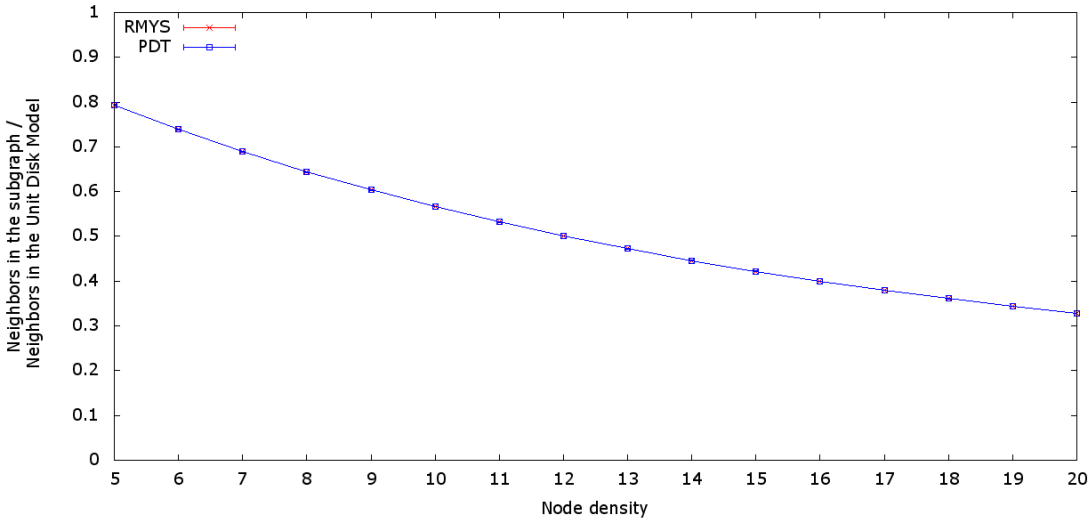

Figure 13: The average neighbors for Reactive Modified Yao Step (RMYS) and Partial Delaunay Triangulation (PDT) divided by the average neighbors in the unit disk model are shown. 1000 Simulations per density.

In Figure 14 is the number of used messages by a topology control divided by the obtained neighbors in the specific subgraph visualized. Here, 1 corresponds to for each neighbor in the subgraph 1 message was sent. PDT uses not much more than this. At density 5 PDT sends approximately 1.33 and at density 20 approximately 1.18 messages per neighbor. PDT on neighbors and RMYS send 6.2 and 6.5, respectively, messages per neighbor at density 5. At density 20 PDT on neighbors uses 8.4 and RMYS 8.6 messages per neighbor.

6 Discussion

In the following the simulation results are interpreted and explained. Additionally, there are two cases where RMYS does not select all PDT neighbors of a node, even if the count of these neighbors is below the limit of $k = 14$. Both cases are presented briefly in the following. At last, it is shown how RMYS handles the nodes if they are positioned in a helix around one node a resulting in arbitrarily many PDT neighbors of a.

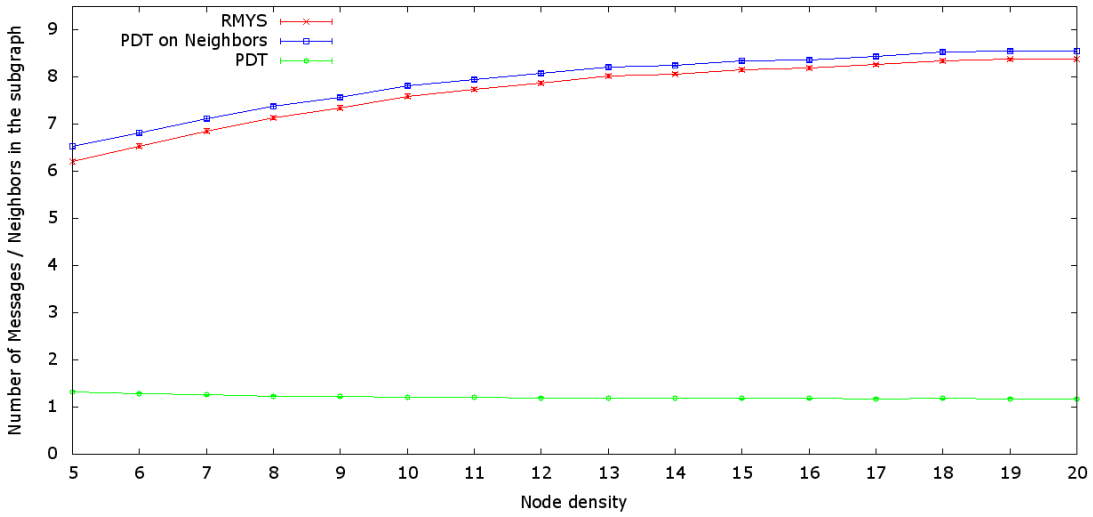

Figure 14: The messages needed to construct the RMYS neighborhood (RMYS), the PDT neighborhood of a node (PDT) and its neighbors (PDT on Neighbors) divided by the actual neighbors in this subgraph are visualized. 1000 Simulations per density.

First, notice figure 15. This figure contains two nodes u and v with one of their RMYS cones drawn. The blue circles and the dotted line marked with R symbolize the unit disk radius of both nodes. The green circle proofs that uv is a PDT edge. Suppose that the red area contains a node n and that any other cone of v (not visualized) also contains one node. If RMYS is executed on that example, u selects v in any case since it is the only node in that cone. However, v selects n since it is closer to v than u. All other cones contain nodes as well and therefore no more edges are selected. As a result of this behavior the edge uv was selected from only one, namely u, of its endpoints and therefore, v sends a protest message.

An example for the second case can be viewed in figure 16. It shows a view on an example graph with all edges drawn which are incident on v. In addition, the Gabriel circles of some nodes are visualized in a dotted way. The green circle through u, v and p proofs that vp is a PDT edge. Note that nodes a, b, c, d and e are also PDT neighbors of v, even though their Gabriel circles and edges to v are not drawn for simplicity of this figure. RMYS executed on node v leads to the selection of all edges incident on v except pv. First, all shortest edges per cone are selected which leads to the selection of uv for the cone containing u and p. Since there is no sequence of empty cones which is longer than 1 only edges which are closest to the empty cones are considered and therefore, pv is not being selected. For this case to happen it is mandatory that there is no sequence of length longer than 1 or the edge is, possibly, being selected. However, it is not needed that

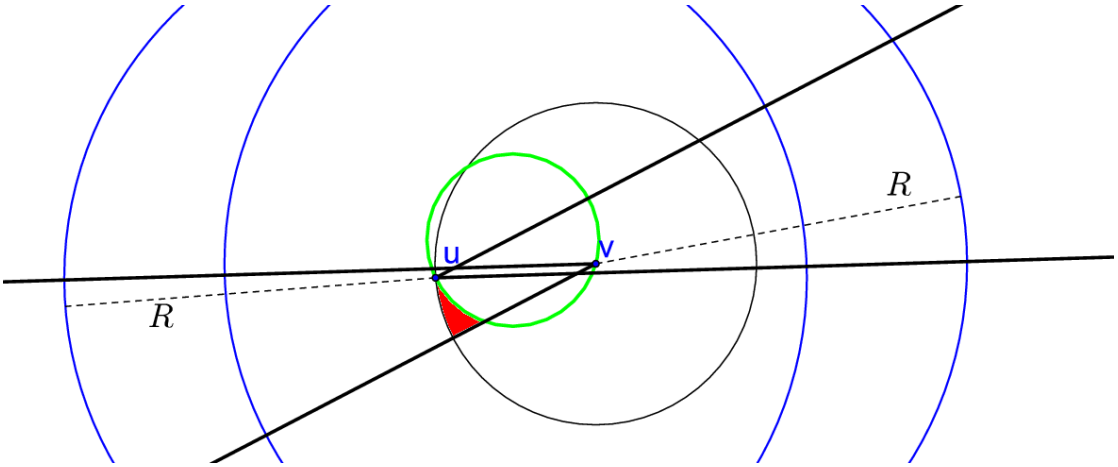

Figure 15: Node u and v are drawn with a specific cone and blue unit disk circles with radius R. The green circle is the PDT circle for edge uv and if the red area contains a point, uv is no bidirectional RMYS edge.

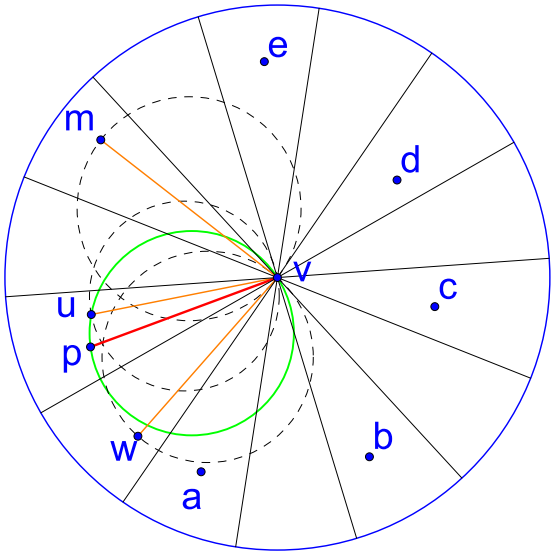

Figure 16: An example graph with specific edges drawn which are incident on v. The blue circle symbolize the unit disk. The dotted circles are Gabriel circles and the green one proofs the existence of uv and pv in the PDT graph.

between nodes a, b, c, d and e is always one empty cone. There can also be none. This is a minimal example in terms of node usage.

In order to understand the small deviation of maximal neighbors at high densities between RMYS and PDT both above explained cases must be taken into account. In addition, the maximal neighbors in figure 12 were calculated over all

nodes in a graph and the actual neighbors for the specific subgraphs were instead calculated for one random node per graph. The small difference of less than 1% between both subgraphs and the fact that no node in the complete simulation has sent a protest message leads to the conclusion that with the parameter $k = 14$ both above cases can be neglected. However, both cases might be relevant for theoretical proofs concerning the spanning ratio or lowering the bound of $k = 14$.

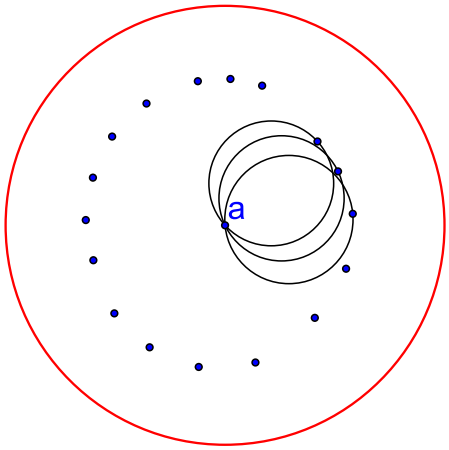

Figure 17: Example point constellation which proofs that PDT not has a constant node degree.

Another interesting aspect to consider is the example which proofs that PDT has a worst case node degree of $\mathcal{O}(n)$. Refer to figure 17 to see an example. The nodes around node a are positioned in a way that every Gabriel circle between a and its neighbors is empty. In other words if you start with node b every next node in clockwise order is a little further away from a. In this case all of these neighbors are PDT neighbors and hence, arbitrarily many nodes can be positioned in this way the worst case node degree of PDT is $\mathcal{O}(n)$. In addition, since all of these neighbors are GG neighbors the node degree of the Gabriel graph is the same. RMYS executed on node a selects for every cone the closest node to a as neighbor and hence, the node degree of this scenario is bounded by $k = 14$ as well.

7 Conclusion

RMYS is due to its reactivity a robust algorithm to construct a bounded degree, planar graph which is most likely an Euclidean spanner. As for now, it is not appropriate to use it to create a local view on just one node in consequence of the message overhead it creates. However, it is well suited to create a complete graph in a reactive way while providing a constant node degree on top of planarity and a constant spanning ratio. RMYS is the first algorithm which inherits all these graph properties and is created in a reactive way.

For the past years reactivity in ad hoc sensor networks reduced permanently the message overhead to create planar graphs which later turned out to be spanners. In addition, it is now possible to append the property of a constant node degree for each node in the graph to reactive approaches. This saves additional messages if a routing protocol uses the underlying RMYS graph.

In order to minimize the constant node degree bound of 14 further research should focus on formally proving the spanning ratio of RMYS and check whether it is possible to lower the degree bound without increasing the spanning ratio significantly. Another interesting point for future research is to reduce the message overhead for RMYS when it is executed on one node. For instance, under the condition that all possibilities where RMYS creates an uni directional edge are known (refer to figures 15, 16 and chapter 6 for help in this matter) one can possibly omit the last broadcast completely. Instead, one must find a pattern for each of these possibilities to detect uni directional edges without sending additional messages.

List of Tables

List of Figures

Bibliography

[1] X. Y. Li, I. Stojmenovic, and Y. Wang, "Partial delaunay triangulation and degree limited localized bluetooth scatternet formation," *IEEE Transactions on Parallel and Distributed Systems*, vol. 15, pp. 350–361, Apr. 2004.

[2] M. Benter, F. Neumann, and H. Frey, "Reactive Planar Spanner Construction in Wireless Ad Hoc and Sensor Networks," in *In Proceedings of 32nd IEEE International Conference on Computer Communications (INFOCOM)*, pp. 2193–2201, Apr. 2013.

[3] S. Rührup, H. Kalosha, A. Nayak, and I. Stojmenović, "Message-Efficient Beaconless Georouting With Guaranteed Delivery in Wireless Sensor, Ad Hoc, and Actuator Networks," *IEEE/ACM Transactions on Networking*, vol. 18, pp. 95–108, Feb. 2010.

[4] M. Chawla, N. Goel, K. Kalaichelvan, A. Nayak, and I. Stojmenović, "Beaconless Position-Based Routing with Guaranteed Delivery for Wireless Ad hoc and Sensor Networks," *Acta Automatica Sinica*, vol. 32, pp. 846–855, Nov. 2006.

[5] F. Neumann and H. Frey, "On the Spanning Ratio of Partial Delaunay Triangulation," in *9th IEEE International Conference on Mobile Ad hoc and Sensor Systems (MASS)*, pp. 434–442, Ieee, Oct. 2012.

[6] M. Damian and S. V. Pemmaraju, "Localized Spanners for Ad Hoc Wireless Networks," *Ad Hoc & Sensor Wireless Networks*, vol. 9, no. 3/4, pp. 305–328, 2010.

[7] I. A. Kanj and G. Xia, "Improved local algorithms for spanner construction," *Theoretical Computer Science*, vol. 453, pp. 54–64, Sept. 2012.

[8] P. Xu, Z. Chen, X. Deng, and J. Yu, "A Partial Unit Delaunay graph with Planar and Spanner for Ad hoc Wireless Networks," *Advanced Materials Research*, vol. 267, pp. 322–327, 2011.

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