<table>
<thead>
<tr>
<th><strong>Title</strong></th>
<th>Finding the polygon hull in wireless sensor networks</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Authors(s)</strong></td>
<td>Bounceur, AHCène; Euler, Reinhardt; Benzerbadj, Ali; Kechadi, Tahar; et al.</td>
</tr>
<tr>
<td><strong>Publication date</strong></td>
<td>2015-06-30</td>
</tr>
<tr>
<td><strong>Conference details</strong></td>
<td>EURO 2015 - 27th European Conference on Operational Research, University of Strathclyde, Glasgow, Scotland, 12-15 July 2015</td>
</tr>
<tr>
<td><strong>Link to online version</strong></td>
<td><a href="http://pagesperso.univ-brest.fr/~bounceur/anr/persepteur/">http://pagesperso.univ-brest.fr/~bounceur/anr/persepteur/</a>; <a href="https://euro2015.euro-online.org/">https://euro2015.euro-online.org/</a></td>
</tr>
<tr>
<td><strong>Item record/more information</strong></td>
<td><a href="http://hdl.handle.net/10197/10844">http://hdl.handle.net/10197/10844</a></td>
</tr>
</tbody>
</table>
Finding a Polygon Hull in Wireless Sensor Networks

Ahcène Bounceur, Reinhardt Euler, Ali Benzerbadj, Farid Lalem, Massinissa Saoudi, Tahar Kechadi and Marc Sevaux
Motivations

- Border nodes
- Radio link
- Non border nodes

Secure site

Intrusion
Dead battery
Interference
Failure
Explosion
Motivations

- Border nodes
- Radio link
- Non border nodes

New border

Secure site

Intrusion
Dead battery
Interference
Failure
Explosion

ALARM
Outline

1. **Polygonal hull in a connected Euclidean graph**
   1. PHFP: Polygon Hull Finding Problem & PHVP: Polygon Hull Visiting Problem
   2. The proposed Algorithm
   3. Simulation: Demo

2. **Material implementation of the algorithm**
   1. The distributed version of the proposed algorithm
   2. Simulation: Demo
   3. Material implementation: Demo

3. **Conclusion & Perspectives**
1.1

PHFP & PHVP
PHFP & PHVP

Convex Hull

Euclidean point set
Convex Hull

Euclidean point set
PHFP & PHVP

Convex Hull

Connected Euclidean Graph
Convex Hull

Connected Euclidean Graph
PHFP & PHVP

Polygon Hull

Connected Euclidean Graph

PHFP

Polygon Hull Finding Problem

Only edges of the graph can be used as connections
PHFP & PHVP

Polygon Hull

*Connected Euclidean Graph*

**PHFP**

*Polygon Hull Visiting Problem*
1.2
THE PROPOSED ALGORITHM
Jarvis’ Algorithm (Euclidean point set)
Jarvis’ Algorithm (Euclidean point set)

\[
\text{pa}(P_a, P_b, P_c) = \text{Angle formed by edges: } P_bP_a \text{ and } P_bP_c
\]

\[
\text{N}(P_c) : \text{set of neighbors of } P_c
\]
Jarvis’ Algorithm (Euclidean point set)

- Complexity: $O(nh)$
- $n$: Number of the nodes of the graph
- $h$: Number of the border nodes
The proposed algorithm (connected graph)
The proposed algorithm (connected graph)
The proposed algorithm (connected graph)
The proposed algorithm (connected graph)

[A, C]
The proposed algorithm (connected graph)

[A, C, D]
The proposed algorithm (connected graph)

[A, C, D, E]
The proposed algorithm (connected graph)

[A, C, D, E, H]
The proposed algorithm (connected graph)

[A, C, D, E, H, G]
The proposed algorithm (connected graph)

[A, C, D, E, H, G, A]

Version 1

2: \( P_c \leftarrow \text{point having the minimum x-coordinate} \)
3: \( B_V \leftarrow [P_c] \)
4: \( P_f \leftarrow P_c \)
5: \( P_p \leftarrow \text{fictive point situated on the left of } P_f \)
6: \( \text{repeat} \)
7: \( P_v = \arg\min_{P_j \in N(P_c)} \{pa(P_p, P_c, P_j)\} \)
8: \( B_V \leftarrow [B_V, P_v] \)
9: \( P_p \leftarrow P_c \)
10: \( P_c \leftarrow P_v \)
11: \( \text{until } P_v = P_f \)
12: \( \text{return } B_V \)
13: \( \text{end procedure} \)
The proposed algorithm (connected graph)

• Version 1
  – Complexity: $O(kh)$
  – $k$: Maximum degree of the graph
  – $h$: Number of the border nodes
PROBLEMATIC SITUATIONS
Boat graph
Boat graph
Boat graph
Boat graph

P -- B -- C -- D

1 (to C)

2
Boat graph
Boat graph

Intersection
Intersection (1)

The currently found border edge must not intersect with the previously found border edges.
Intersection (2.1) Consider intersections with open intervals

\( ]a, b[ \cap ]c, d[ \) and not: \( [a, b] \cap [c, d] \)
Anchor graph
Anchor graph

Formed by the edge AC and the triangle BCD
Graphe à Ancre

Two angles (PBC et PBD) having the same value → Choose the point which is near to B (ie. C)
Anchor graph

If the algorithm starts from A then the point A will be reached but not the point J!
Anchor graph

If the algorithm starts from A then the point A will be reached but not the point J!
Anchor graph

PHFP : solved?
PHVP : solved?
Anchor graph

PHFP : solved?
PHVP : solved?
Anchor graph

Coming back to the starting point does not guarantee the termination of the algorithm!
→ We have to add an additional iteration to verify if there exists a non visited border node.
The proposed algorithm (Version 2)

2: \( P_c \leftarrow \text{point having the minimum x-coordinate} \)
3: \( \mathbb{B}_V \leftarrow [P_c] \)
4: \( P_f \leftarrow P_c \)
5: \( P_p \leftarrow \text{fictive point situated in the left of } P_f \)
6: \( \text{repeat} \)
7: \( \mathbb{A} = \emptyset \)
8: \( P_v = \underset{P_j \in N(P_c) \& P_j \notin \mathbb{A}}{\text{argmin}} \{ \text{pa}(P_p, P_c, P_j) \} \)
9: \( \text{if intersection detected then} \)
10: \( \mathbb{A} = \mathbb{A} \cup \{ P_v \} \)
11: \( \text{Go to 8} \)
12: \( \text{end if} \)
13: \( \mathbb{B}_V \leftarrow [\mathbb{B}_V, P_v] \)
14: \( P_p \leftarrow P_c \)
15: \( P_c \leftarrow P_v \)
16: \( \text{until } P_v = P_f \)
17: \( \text{return } \mathbb{B}_V \)
18: \( \text{end procedure} \)
Proposed algorithm (connected graph)

• Version 1 (planar graph)
  – Complexity: $O(kh)$
  – $k$: Maximum degree of the graph
  – $h$: Number of the border nodes

• Version 2
  – Complexity: $O(kh^2)$
  – $k$: Maximum degree of the graph
  – $h(h+1)/2$: Number of the border nodes by considering the intersection detection
1.3

SIMULATION: DEMO
CupCarbon: www.cupcarbon.com

- A WSN design and simulation platform
- Java API (Application Programming Interface)

→ VIZOR
Simulation: Demo

https://www.youtube.com/watch?v=7VHKW_I9jul
Simulation: Demo (100 points)
Simulation: Demo (500 points)
Simulation: Demo (500 points)
Simulation: Demo (1500 points)
2.1 DISTRIBUTED VERSION OF THE PROPOSED ALGORITHM
Sensor node

1. Sensor (Sensor unit)
2. Radio module (communication)
3. Microcontroller $\leftarrow$ Script (program)
4. Battery
5. GPS
6. Mobility
Messages & Packets

• 3 types of messages:
  AC (Ask for Coordinates): Send me your coordinates
  CS (Coordinate Sending): My coordinates are
  SN (Select Node): You are a border node

• Packets format:
  Message AC: id | AC
  Message CS: id | CS | x,y
  Message SN: id | SN | x,y
The distributed version of the proposed algorithm

S1 is marked as a border node
The distributed version of the proposed algorithm

S1 send a broadcast message ‘AC’ and its id
The distributed version of the proposed algorithm

S2, S3, S4 and S7 receive the message ‘AC’ and send to S1 their coordinates, their id and the message ‘CS’
The distributed version of the proposed algorithm

**S1 will calculate the angles formed by a fictive point S1’ and its neighbors and chooses the smallest one.**
The distributed version of the proposed algorithm

S1 will send a message ‘SN’ to S3 to inform it that it is the next border sensor, its id and its coordinates.
The distributed version of the proposed algorithm

S1 saves the id of its neighbor S3
The distributed version of the proposed algorithm

S1 \rightarrow S3
S7 \rightarrow S1
S3 \rightarrow S4
S4 \rightarrow S5
S5 \rightarrow S8
S8 \rightarrow S7
2.2

SIMULATION: DEMO
Simulation: Demo

https://www.youtube.com/watch?v=ojgZPlvwcMo
2.3

MATERIAL IMPLÉMENTATION: DEMO
Implementation: Demo

https://www.youtube.com/watch?v=vObmCb67jg4

Sensors: TelosB / Arduino-XBee
CONCLUSION & PERSPECTIVES
Conclusion & Perspectives

• Presentation of a new algorithm allowing to determine the polygon hull of a wireless sensor network:
  – Sequential version: allows to validate the theoretical concept of the algorithm
  – Distributed version: allows the real implementation
• Model the same problem by taking into account different parameters: detection, energy, interferences, etc.
• We are working on:
  – The GPU based simulation in CupCarbon
  – Energy Consumption of the proposed algorithm
  – Proving the optimality in number of vertices and total distance of the proposed algorithm
Thank you for your attention

Questions?

Ahcène Bounceur, Reinhardt Euler, Ali Benzerbadj, Farid Lalem, Massinissa Saoudi, Tahar Kechadi and Marc Sevaux

Ahcene.Bounceur@univ-brest.fr
www.bounceur.com / www.cupcarbon.com