Early Obstacle Detection and Avoidance for All to All Traffic Pattern in Wireless Sensor Networks

Huc, Florian; Jarry, Aubin; Leone, Pierre; Moraru, Luminita; Nikoletseas, Sotiris; Rolim, Jose

How to cite

HUC, Florian et al. Early Obstacle Detection and Avoidance for All to All Traffic Pattern in Wireless Sensor Networks. In: The proceedings of the 5th International Workshop on Algorithmic Aspects of Wireless Sensor Networks, ALGOSENSORS 2009. Rhodes (Greece). Berlin, Heidelberg : Springer, 2009. (Lecture Notes in Computer Science) doi: 10.1007/978-3-642-05434-1_11

This publication URL: https://archive-ouverte.unige.ch//unige:32658
Publication DOI: 10.1007/978-3-642-05434-1_11

Early Obstacle Detection and Avoidance for All to All Traffic Pattern in Wireless Sensor Networks

Florian Huc[1], Aubin Jarry[1], Pierre Leone[1], Luminita Moraru[1] ⋆, Sotiris Nikoletseas[2], and Jose Rolim[1]

[1] Computer Science Department
University of Geneva
1211 Geneva 4,Switzerland
[2] University of Patras and CTI
26500 Patras, Greece

Abstract This paper deals with early obstacles recognition in wireless sensor networks under various traffic patterns. In the presence of obstacles, the efficiency of routing algorithms is increased by voluntarily avoiding some regions in the vicinity of obstacles, areas which we call dead-ends. In this paper, we first propose a fast convergent routing algorithm with proactive dead-ends detection together with a formal definition and description of dead-ends. Secondly, we present a generalization of this algorithm which improves performances in all to many and all to all traffic patterns. In a third part we prove that this algorithm is optimal up to a constant factor as we prove it computes path whose length is at most $2\pi + 1$ times longer than the shortest path. In a fourth part we consider the reactive version of the algorithm which is an extension of a previously known early obstacle detection algorithm. Finally we give experimental results to illustrate the efficiency of our algorithms in different scenarios.

1 Introduction

In this paper, we study the problem of routing messages in sensor networks. Due to the specificity of such networks, such an algorithm needs to made computations based on local information (information that can be found on nodes that are close). Numerous algorithms have been proposed. For a class of them which is called geographic routing algorithms, the hypothesis that the sensors know their geographic position is made (for instance using a GPS). The algorithms we propose in this paper belong to this class.

Geographic routing algorithms differ in the way they avoid obstacles (Definition 1). Indeed, the strategy of these algorithms is mainly to greedily progress towards the destination (whose position is supposed to be known). To do so, the nodes coordinates are used to compute the euclidean distance to the destination, and the greedy forwarding consists in chosing one of the node whose euclidian distance to the destination is the smallest. But obstacles can block such computed path in a local minima. Hence several techniques have been introduced to get out of local minima ; for instance, GRIC follows the obstacles' borders while introducing some inertia ; Face Routing planarizes the network and then routes along faces. These algorithms do not use other information than the coordinates of the nodes and their efficiency is usually estimated according to the amount of computation needed at each node and the length of the path discovered in comparison to the length of the shortest path d. The algorithm presented in [KWZ08] discovers a path whose length is at most $O(d^2)$; furthermore it is shown that no memoryless algorithm can guarantee a better result.

⋆ Research partially funded by FP6-015964 AEOLUS

To improve path discovery, some algorithms using extra node memory have been proposed. This is the case of the early detection algorithm proposed in [MLNR07]. We investigate further this solution and we propose a class of proactive obstacle detection algorithms (ODA). [MLNR07] considers networks containing only one base station, with a *all to one* data traffic pattern. In this paper we propose a faster convergent algorithm which works under the same hypothesis but also in the case of all to many and all to all traffic pattern hypothesis. We then propose an extension that improves its performances in the case of *all to many* and *all to all* traffic pattern.

These algorithms mark the nodes in the vicinity of obstacles while routing messages, and favor the non marked ones. One of the performance parameters of ODA is the time of convergence - the time needed to acknowledge the presence of obstacles in their vicinities. An important contribution of this paper is a significant improvement of the convergence time in comparison to [MLNR07]. Furthermore we prove that this algorithm computes a path whose length is at most a constant times the length of the shortest path (and hence optimal up to a constant factor) under the hypothesis that the plane is covered uniformly by sensor except on the obstacles ; this situation being an approximation of a dense sensor network. To do so, we give a formal definition of marked area, called dead-end areas, and give a description of them.

The paper is organized as follow: Section 2 introduces the current state of the art in geographic routing, with emphasis on obstacle avoidance algorithms. Section 3 contains the description of our fast convergent and proactive algorithm for all to one traffic pattern (remarks that this algorithm also work under other traffic patterns). Section 4 describes improvements of this algorithm for all to many and all to all traffic patterns. Section 5 is an analysis of dead-ends (or marked) areas and it includes the proof that our algorithm computes a path whose length is at most $(2\pi + 1)d$, where d is the length of the shortest path. In Section 6, we propose the reactive version of the algorithm. In Section 7, we compare the performances of our algorithms with the state of the art. The conclusions are presented in the last section.

2 State of the art

We address the problem of finding near optimal paths in sensor networks using geographic routing. We suppose that all nodes know their position and that obstacles are present in the network. Considering this framework, several scenarios can be considered : *all to one*, when any node in the network can be the originator of a message whose destination is a unique Base Station; *all to many* when several Base Stations are present in the network ; *all to all* when any node can be destination of a message.

By default, geographic routing algorithms greedily route the messages towards the destination [KSU99], [Fin], [MM96], [CNS05]. When the greedy forwarding technique fails, the algorithms enter a recovery mode, used until the greedy forwarding technique is again feasible. The solutions are divided in the following categories[CV07]: flood based, planar graph based, spanning tree based and geometric based.

Flooding based techniques [SL01],[JPS99] broadcast the message if it reaches a local minima. Although the complexity is low, the overhead is high. And, even if they guarantee delivery, path optimality is not a concern. An alternative to flooding is multicast. [CDNS06] proposes a redundant multipath delivery scheme that uses probabilistic choices to achieve an optimal trade-off between efficiency and cost; while this method indeed copes well in sparse networks, it fails to bypass big obstacles.

[BCN06] is a protocol combining greedy routing and adaptation of the transmission range to bypass obstacles. It manages to "jump over" obstacles, but the routing path created is not optimal and the energy cost can become high.

Planar graph traversal techniques [KK00], [HBBW03], [BMSU01], [DSW02] are used since they were proved to guarantee delivery if a path exists. Planar graph based obstacle avoidance strategies use greedy as long as a node has a neighbor closer to the destination. Otherwise, one of the existing planar graph traversal algorithms [Urr02], [KWZ03], [KWZZ03] is used. Since the representation of the network is not always a planar graph, this class of strategies needs to use a distributed planarization algorithm. This can be done at the network level, in the network setup phase, or it can be done on demand, only for the set of nodes where greedy forwarding cannot be used.

The performances of these strategies depend on two factors: the performances of graph traversal algorithm and of the distributed planarization algorithm [KGKS05], [GS69], [Tou91]. In [KWZ08], the authors propose a variant of [KK00] which computes paths whose length is $O(d^2)$, where d is the length of the shortest paths to the destination. Furthermore, they prove that no algorithm not using sensors's memories can guarantee a better approximation ratio.

Spanning tree based techniques build a spanning tree when a message is blocked at a node. In [RRR$^+$99] they are forwarded using flooding, while in [LLM06] the locations covered by subtrees are aggregated using convex hulls to decide which direction in the tree is closer to the destination. The disadvantage of the method is the overhead needed to transmit the information several hops away.

Geometric based techniques use memories of sensors to capture geometric properties of the network in order to improve performances. For instance, a geometric obstacle avoidance algorithm is proposed in [FGG06]. It uses the geometric properties of a node (position of its neighbors) to determine if a message can be blocked at that node. The objective is to find obstacles of the network, which are areas of the network bounded by nodes at which messages are blocked. This technique needs angle computation and hence the sensors are supposed to have specific equipment. Furthermore, the obstacle detection is done during an initialisation phase and hence decrease the available energy of all nodes, even the one that are not in a local minima.

Other existing early obstacle detection techniques using geographic routing where concerned with the all to one scenario, i.e. when all the traffic was directed towards a single base station. They are based on the detection, and marking of nodes close to obstacles in a reactive manner. This way it avoids the use of energy at nodes which are not local minima.

The algorithms proposed in [MLNR07,MLNR08] are based on two modes, the greedy mode and the perimeter mode. In greedy mode, the node which has to send a message choose among its optimal (see bellow) neighbors the one that decrease the most the euclidien distance towards the destination. When a node is in perimeter mode, it means that the message is currently following the border of an obstacle in some direction. The node sends the message so that it keeps following the border of the obstacle in the same direction, until it is completely avoided, see [KK00] for a precise description of this mode.

The two algorithms differ in the way to mark optimal or non optimal nodes. Initially all nodes are marked as optimal. The method [MLNR07]: *behavior based routing* evaluates the ratio between the total number of times greedy or perimeter routing were previously used by a node. If the ratio is in favour of perimeter, the node is marked as non optimal. In the second method [MLNR08]: *neighborhood based routing*, a node on a routing path tags itself based on the outcome of a node optimality evaluation method - the evaluation is optimal if a node has at least one neighbor tagged as optimal and which is closer to the destination. If a node is non-optimal, then we consider that any path towards the destination using it will be as well non-optimal. These protocols gradually evaluates the performance of a path, detecting reactively the nodes around obstacles, and progressively redefining the routing paths. Each node is evaluating itself and spreads locally information about its performances. Once the non optimal nodes are detected

and advertised, each node in greedy routing mode will avoid to choose non optimal neighbors for forwarding, thus redirecting the message outside the non optimal area. There are two main disadvantages of these methods. First, the convergence time is slow since several paths needs to be routed through non optimal areas before they are detected. Second, it lacks generality. Indeed, if there are several base stations in the network, then the paths could be even longer than with the classical algorithms (e.g. GPG).

3 Single destination routing and *dead-ends* detection

To avoid *local minima*, a solution consisting in marking nodes with optimal and non-optimal reputations has been developed in [MLNR07]: non-optimal nodes lead to *local minima* and should be avoided, whereas nodes marked as optimal can be safely used for greedy routing.

Similarly, in this paper we will consider marking nodes as optimal or non-optimal. Our marking bears important differences as it is done proactively whenever a local minima is detected.

Initially, all nodes are marked as optimal. Then the following algorithm is run, in which *current node* stands for any node that has a message to send.

Algorithm 1 Algorithm Dead-End

```
Input : A node and the destination of its message.
Output : A node to which to forward the message.
The current node proceeds to a dead-end evaluation process;
if The current node is optimal then
   it chooses the next node using a greedy mode;
else
   it chooses the next node using an escape mode.
end if
```

Dead-end evaluation process The current node checks if one of its neighbors is closer to the destination. If not, it is a local minima towards the destination. If it is not a local minima, it makes nothing, otherwise it means it has no neighbors closer to the destination. In this case, it marks itself as non-optimal and broadcasts this information to all its neighbors. Each node which receives such a message checks if it has *optimal* neighbors closer to the destination than itself. If not, it marks itself as non-optimal and broadcast it. The same process repeats recursively until a whole area is marked as non-optimal. We call such an area a *dead-end*.

At this point, an exit is computed for the messages originated at non-optimal nodes. When, during the evaluation process, a node evaluates itself as optimal, it advertises that it is a potential exit from the dead-end. Then a non-optimal node knowing an exit, advertises itself as an exit recursively. Each non-optimal node chooses as exit the first neighbor advertised as exit.

Remark 1. This algorithm induces communications only in dead-ends. Hence any node outside a dead-end will NOT waste energy. Also nodes in dead-ends use energy at this point to mark themselves so that not to be contacted later on. Finally, the overhead is of exactly two broadcasts per node in the dead-end and one for the nodes adjacent to one in the dead-end. Hence this algorithm would be efficient in large sensor networks.

Greedy mode Good nodes function in greedy mode. They route received messages greedily, i.e., they send the message to the optimal node among their neighbors which is the closest to the destination.

Escape mode Bad nodes function in escape mode. They know (cf dead-end process above) one of their neighbors as a direction towards the exit of the *dead-end*, so they send messages to this node.

Algorithm convergence: we say that the algorithm converges when the number of non-optimal nodes does not increase anymore. It means, that a message send by an optimal node will be entirely routed greedily, avoiding any *dead-ends*. A message send by a non-optimal node will first escape the dead-end until it reaches an optimal node, after what it will be routed greedily.

4 Multiple directions routing and *dead-ends* detection

The previous algorithm detects the presence of obstacles towards the destination of a message, and marks nodes accordingly. It increases the performances of the routing algorithm for messages sent to this destination, but, if there are multiple destinations, it can decrease the performances for a message directed to another destination. We are aiming at finding efficient paths regardless of the destination of the message. A solution is to divide the communication area reached by the node in several directions and to keep track of the optimality for each direction. A parameter *directions* is chosen accordingly to the number of destinations, if there are several base station for instance, or accordingly to the shape of the obstacles if they are known. Otherwise, an arbitrary size can be chosen (heuristically height works well). To each node is associated an array of size *directions*, which indicates if the node is optimal for a specific direction or not.

Given a node n, and a message that it has to send, n computes in which direction i the destination is. For example, if there are four directions, a node n, split the networks into the four geometrical quadrants.

The algorithm is an adaption of the one of the previous section:

Algorithm 2 Algorithm Directional Dead-End

```
Input : A node and the destination of its message.
Output : A node to which to forward the message.
The current node computes in which direction is the destination;
The current node proceeds to a dead-end evaluation process for this direction;
if The current node is an optimal node for this direction then
    it chooses the next node using a greedy mode;
else
    it chooses the next node using an escape mode.
end if
```

Each of the *greedy mode, escape mode* and *dead-end process* that we compared are described in the following.

Greedy mode If node n is marked as optimal for direction i, it chooses among all its neighbors that are closer to the destination and marked as optimal for the direction of the destination, the closest to the destination (notice that from the neighbor point of view, the direction towards the destination is not necessary i).

Escape mode As in the algorithm of Section , if a node is marked as non-optimal, it has precomputed a neighbor which he knows to be an exit for the *dead-end* in direction i. So the node sends the message to this precomputed node. The dead-end exit is relative to a direction and hence can be different for each direction.

Dead-end evaluation process This process verifies locally if a greedy routing would find a node marked as optimal. If so it does nothing, otherwise it do what is described in section 4, except that the notion of optimal and non-optimal node is relative to the direction of the destination, let say i. To summarize, the process is the following: the current node marks itself as non-optimal for direction i and broadcast the information to all its neighbors. Any other node that marks itself as non-optimal also broadcast the information and the process is repeated recursively until it stops. When this process end, the exit of the dead-end area is computed, c.f. Section 4. And a node is marked as non-optimal for the direction of the destination i, if all its neighbors closer to the destination are marked as non-optimal for this direction i.

5 Dead-end analysis

This section aims at describing from a theoretical point of view the behaviour of our algorithms using the *recursive execution of the dead-end evaluation process*. We indicate which areas (called *dead-ends*) need to be marked such that a message routed greedily avoid all local minima.

We concentrate ourselves in the case of all to one routing pattern as this analyse extends to the all to many routing pattern.

To describe which part of the network is marked as non-optimal, we use a *continuous model*. We suppose that the plane is completely covered by sensors except on some regions called obstacles (definition 1 below), that prevent direct communications between nodes. *Each point of the plan that is not an obstacle represent a node that can transmit a message.*

Theorem 1 describes the minimum part of the plane that will be marked as a dead-end area when there is a single destination D.

Definition 1 (obstacle). *An obstacle p is a position in the plane such that for any pair of nodes (x_1, x_2) at positions p_1 and p_2, if $p \in [p_1, p_2]$ then x_2 is not a neighbor of x_1.*

We use polar coordinates with origin the destination D, to describe the positions of nodes in the network.

Theorem 1. *If there is a continuous line of obstacles $\{(f(\alpha), \alpha)\}_{\alpha \in [\beta,\gamma]}$ such that $f : [\beta, \gamma] \to \mathbb{R}$ is a continuous function, then the dead-end area contains at least all the sensors at position (ρ, α) such that $\alpha \in [\beta, \gamma]$ and $f(\alpha) < \rho \leq \min(f(\beta), f(\gamma))$.*

Proof. Let x_1 be a sensor that is not in a dead-end area. There is a sequence of nodes $x_2, \dots x_n$ such that $x_n = O$, and $\forall 0 < i < n$ x_i is a neighbor of x_{i+1} and x_{i+1} is closer to the destination than x_i.

Suppose that x_1 is at position (ρ_1, α_1) with $\alpha_1 \in [\beta, \gamma]$ and $f(\alpha_1) < \rho_1 \leq \min(f(\beta), f(\gamma))$. Then there is a node x_i at position (ρ_i, α_i) such that

- $\alpha_i \in [\beta, \gamma]$ and
- $f(\alpha_i) < \rho_i \leq \min(f(\beta), f(\gamma))$ and
- x_{i+1} is at position $(\rho_{i+1}, \alpha_{i+1})$ with
  - $\min(f(\beta), f(\gamma)) < \rho_{i+1}$ or
  - $\alpha_{i+1} \notin [\beta, \gamma]$ or
  - $\rho_{i+1} \leq f(\alpha_{i+1})$.

Since x_{i+1} is closer to D than x_i, we have $\rho_{i+1} < \rho_i$, and therefore $\rho_{i+1} \leq \min(f(\beta), f(\gamma))$. If $\alpha_{i+1} \notin [\beta, \gamma]$ or if $\rho_{i+1} \leq f(\alpha_{i+1})$, then $[(\rho_i, \alpha_i), (\rho_{i+1}, alpha_{i+1})]$ intersects the line $\{(f(\alpha), \alpha)\}_{\alpha \in [\beta,\gamma]}$ and x_{i+1} can not be a neighbor of x_i.

Remark 2. The analysis works similarly if we consider several directions. The difference is that there is multiple dead-ends, each being associated to a given direction. A node can be in as many dead-ends as there are directions.

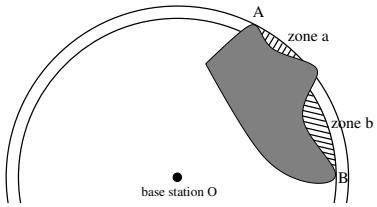
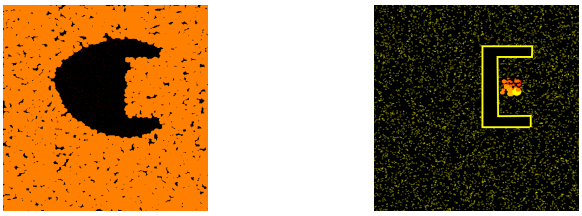


Figure 1. A line of obstacles from A to B creates dead-ends : zone a and zone b.



(a) Dead-end (b) Obstacle

Figure 2. Simulations show Dead-ends.

Approximation ratio

In this section we consider a network with a single obstacle, however there is no hypothesis on the shape and size of the obstacle.

We consider a message whose origin and destination are outside the obstacle and dead-ends. Our objective is to prove that, in a continuous model, the path computed by our algorithm is at most $2\pi + 1$ times longer (in terms of distance) than the shortest path, and therefore optimal up to a constant factor.

Definition 2. *Given a point O and an obstacle, the apparent angle of the obstacle is the angle of the smallest circular sector originating in O and containing the obstacle.*

Theorem 2. *Given a network with a single obstacle under the continuous model assumption and a message with source s and destination t, when the corresponding dead-ends are marked and if s is not situated in the dead-ends, calling α the apparent angle of the obstacle from t, if $\alpha < 2\pi$, then the length of the path computed by the algorithm is at most $(\alpha + 1)d$, where d is the length of the shortest path from s to t.*

Proof. The dead-ends are described by Theorem 1. The algorithm does not enter dead-ends by definition. Instead, it follows their border. The worst case happens when the computed path bypass the obstacle from the other side than the shortest path. In this case, the computed path will turn around the whole obstacle following the dead-ends border (Notice that the distance to the destination may not increase since we are in greedy mode). The path will at most turn around the obstacle for an angle of α ,the apparent angle of the obstacle from t, before finding a straight line to the destination ($\alpha < 2\pi$ by hypothesis). Hence the computed path has length at most $(\alpha + 1)d(s, t)$, where $d(s, t)$ is the distance between s and t. Finally, the shortest path has length d at least $d(s, t)$. Therefore, the computed path is at most $(\alpha + 1)$ times longer than the shortest path.

6 Reactive Dead-end Discovery

In this section, we propose reactive variants of the algorithm proposed in Section 4 : Directional Dead-End. By reactive, we mean that only the node forwarding the message computes wether it is optimal or non-optimal. This algorithm combines the multiple marking according to a number of chosen directions, and the caracteristics of the algorithms presented in [MLNR07,MLNR08]. They converge slower than the ones proposed in Sections 3 and 4 but the global overhead is reduced.

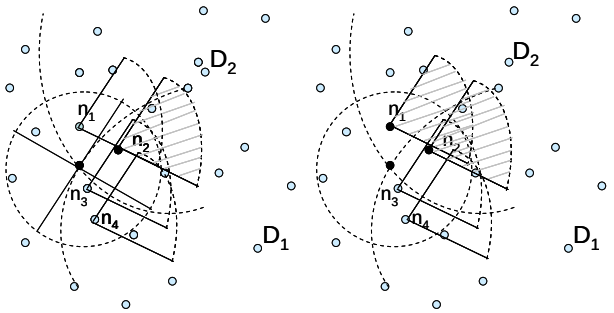
As before, there are two modes, the greedy mode during which the next node is chosen greedily among the optimal neighbors, and the perimeter mode.

perimeter mode: This is the routing mode inside dead-ends. We use GPSR, it works on a planarized graph of the network and routes the message around a face towards the destination. This algorithm is explained in the introduction.

When constructing a path to route a message, nodes along the path are evaluated. The current node design the node which currently has the message to send. In this Section, only the current node is evaluated at each time step.

We compare two ways to determine whether a node is optimal or non-optimal.

greedy neighborhood evaluation: In this evaluation, we consider all the neighbors closer to the destination than the current node. A node marks itself as non-optimal for the direction in which is the destination if all of them are non-optimal for this direction and optimal for this direction otherwise. Notice that the number of non-optimal node may either increase or decrease in the network as we consider traffic towards multiple directions, this is illustrated by figure 3.



(a) network context 1 (b) network context 2

Figure 3. example.

First we describe the fig. 3. The current node is situated at the center of the circle and it has to send messages towards the two destination d_1 and d_2. For the sake of simplicity the area covered by the node is divided in only four directions. For each neighbor only the direction towards the destination is taken into account for this evaluation. The directions used for the evaluation are marked in the two pictures. We use an empty segment for optimal nodes and a segment with stripes for non-optimal nods: n_1 and n_2 in the first picture and n_2 in the second picture. There are two greedy nodes, n_1 and n_2 towards d_2 and three greedy nodes, n_2,n_3,n_4 towards d_1.

In this example, the greedy neighborhood evaluation works as follow : in the first picture, the current node contains optimal neighbors closer to both destination (n_3 for d_1, and n_1 for d_2), therefore it will be marked as optimal for this direction. In the second picture, all neighbors closer to d_2 are supposed to be non-optimal, so the current node will mark itself as non-optimal for this direction. However, if an evaluation is made towards d_1, which is in the same direction, the current node will mark itself as optimal, whatever is previous mark was.

one flip evaluation: A drawback of the previous method is that nodes will often change their mark, depending on the message destination. This will trigger additional traffic in the network. In one flip evaluation, we keep non-optimal nodes. In other words, once a node is marked as non-optimal, it remains non-optimal. In this evaluation method, the number of non-optimal nodes can only increase.

In the second example of fig. 3, once the current node has been evaluated towards d_2, it is marked as non-optimal and remains non-optimal, even if it routes a message towards d_1.

7 Simulation results

The measurements are made with a network of 50x50 with a single obstacle. The network traffic is from left to right. We are using different network densities, from 20 to 40 neighbors in average.

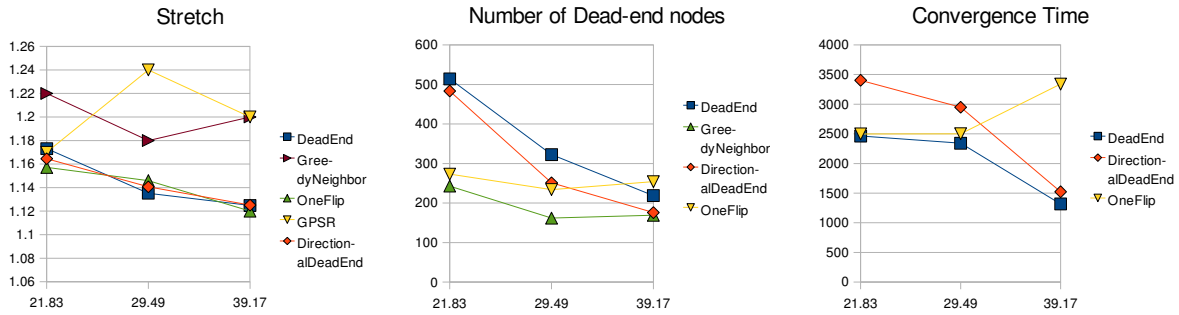


Figure 4. Simulation results : path stretch, number of dead-end nodes and convergence time

7.1 Stretch

In this section, we compare the average ratio between the path computed by various algorithms and the distance between the source and the destination. The algorithm compared are GPSR which serves as point of comparison with existing work, Dead-End, Directional Dead-End, greedy neighbor evaluation and one flip evaluation.

The obstacle is a half moon situated in the middle of the network, the network traffic is all to all.

We observe that the algorithms Dead-End, Directional Dead-End and one flip evaluation perform well and computes close to optimal path in terms of path length. The improvement consisting in marking nodes using a direction is not significant (Dead-End performs as well as the two others), especially when compared with the increased complexity : it needs more computation at each steps and more memories at each nodes.

7.2 Dead-end nodes

The number of dead-end nodes decrease when the density of the network increases. This can be explained by the fact that when the density increase, the dead-ends converge to the dead-ends described in Section 5. The area covered by dead-end nodes decreases more rapidly than the density increases.

7.3 Convergence time

We compare the convergence time of the various algorithms. For Dead-end and Directional Dead-end, fig. 4 indicates the time after which no more nodes have been marked during 400 steps. One flip evaluation does not converge in this sense, however, fig. 4 indicates the time after which no more nodes have been marked during 200 steps ; although, if the number of non-optimal node after this time is not stable, it does not change much. Greedy neighbor evaluation is not convergent even in this sense. Therefore, Dead-end and Directional Dead-end are more efficient algorithms in terms of convergence.

The convergence time decreases when the density increases, this corroborates the decreasing number of dead-end nodes when the density increases. Indeed it takes less time to mark all of them.

8 Conclusion

In this paper we introduced several improvements to the obstacle detection and avoidance routing algorithms and compared them together. We showed that our dead-end recursive evaluation technique outperforms previous algorithms in terms of convergence time. The algorithms proposed all do well in terms of path stretch. The main contribution is a theoretical guaranty that, in a continuous network, the algorithm Dead-End computes a path whose length is at most $\pi+1$ times bigger than the length of the shortest path. This result is interesting since it is optimal up to a constant factor and that, without using memories, the best distributed algorithm cannot guarantee to compute a path whose length is smaller than the square of the distance.

However, the algorithms we proposed are designed for static networks. It would be interesting to extend them to the dynamic case in the hope of getting theoretical guaranties on the computed path under realistic hypothesis on the dynamicity of the system. An approach may consist in checking the non-optimality of non-optimal nodes after a certain elapsed time.

References

[BCN06] Azzedine Boukerche, Ioannis Chatzigiannakis, and Sotiris E. Nikoletseas. A new energy efficient and fault-tolerant protocol for data propagation in smart dust networks using varying transmission range. *Computer Communications (COMCOM) Journal*, 29(4):477–489, 2006.

[BMSU01] Prosenjit Bose, Pat Morin, Ivan Stojmenović, and Jorge Urrutia. Routing with guaranteed delivery in ad hoc wireless networks. *Wirel. Netw.*, 7(6):609–616, 2001.

[CDNS06] I. Chatzigiannakis, T. Dimitriou, S. Nikoletseas, and P. Spirakis. A probabilistic algorithm for efficient and robust data propagation in smart dust networks. *Ad-Hoc Networks Journal*, 4(5):621–635, 2006.

[CNS05] Ioannis Chatzigiannakis, Sotiris Nikoletseas, and Paul G. Spirakis. Efficient and robust protocols for local detection and propagation in smart dust networks. *Special Issue on Algorithmic Solutions for Wireless,Mobile, Ad Hoc and Sensor Networks, ACM/Baltzer Mobile Networks and Applications(MONET) Journal*, 10(1-2):133–149, 2005.

[CV07] D. Chen and P.K. Varshney. A survey of void handling techniques for geographic routing in wireless networks. *Communications Surveys and Tutorials, IEEE*, pages 50–67, 2007.

[DSW02] Susanta Datta, Ivan Stojmenovic, and Jie Wu. Internal node and shortcut based routing with guaranteed delivery in wireless networks. *Cluster Computing*, 5(2):169–178, 2002.

[FGG06] Qing Fang, Jie Gao, and Leonidas J. Guibas. Locating and bypassing holes in sensor networks. *Mob. Netw. Appl.*, 11(2):187–200, 2006.

[Fin] GG Finn. Routing and addressing problems in large metropolitan-scale internetworks, ISI Research Report ISU. Technical report, RR-87-180, March 1987.

[GS69] K.R. Gabriel and R. R. Sokal. A new statistical approach to geographic variation analysis, 1969.

[HBBW03] M. Heissenbüttel, T. Braun, T. Bernoulli, and M. Wälchli. BLR: Beacon-less routing algorithm for mobile ad-hoc networks, 2003.

[JPS99] R. Jain, A. Puri, and R. Sengupta. Geographical routing using partial information for wireless ad hoc networks, 1999.

[KGKS05] Young-Jin Kim, Ramesh Govindan, Brad Karp, and Scott Shenker. Geographic routing made practical. In *NSDI'05: Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation*, pages 16–16, Berkeley, CA, USA, 2005. USENIX Association.

[KK00] Brad Karp and H. T. Kung. GPSR: greedy perimeter stateless routing for wireless networks. In *Mobile Computing and Networking*, pages 243–254, 2000.

[KSU99] Evangelos Kranakis, Harvinder Singh, and Jorge Urrutia. Compass routing on geometric networks. In *Proc. 11 th Canadian Conference on Computational Geometry*, pages 51–54, Vancouver, August 1999.

[KWZ03] Fabian Kuhn, Roger Wattenhofer, and Aaron Zollinger. Worst-Case Optimal and Average-Case Efficient Geometric Ad-Hoc Routing. In *Proc. 4^{th} ACM Int. Symposium on Mobile Ad-Hoc Networking and Computing (MobiHoc)*, 2003.

[KWZ08] F. Kuhn, R. Wattenhofer, and A. Zollinger. An algorithmic approach to geographic routing in ad hoc and sensor networks. *IEEE/ACM Transactions on Networking*, 16(1):51–62, 2008.

[KWZZ03] F. Kuhn, R. Wattenhofer, Y. Zhang, and A. Zollinger. Geometric ad-hoc routing: Of theory and practice, 2003.

[LLM06] Ben Leong, Barbara Liskov, and Robert Morris. Geographic routing without planarization. In *NSDI'06: Proceedings of the 3rd conference on 3rd Symposium on Networked Systems Design & Implementation*, pages 25–25, Berkeley, CA, USA, 2006. USENIX Association.

[MLNR07] Luminita Moraru, Pierre Leone, Sotiris Nikoletseas, and José D. P. Rolim. Near optimal geographic routing with obstacle avoidance in wireless sensor networks by fast-converging trust-based algorithms. In *Q2SWinet '07: Proceedings of the 3rd ACM Workshop on QoS and security for wireless and mobile networks*, pages 31–38, New York, NY, USA, 2007. ACM.

[MLNR08] L. Moraru, P. Leone, S. Nikoletseas, and J. Rolim. Geographic Routing with Early Obstacles Detection and Avoidance in Dense Wireless Sensor Networks. *Lecture Notes in Computer Science*, 5198:148–161, 2008.

[MM96] Rudolf Mathar and Jürgen Mattfeldt. Optimal transmission ranges for mobile communication in linear multihop packet radio networks. *Wirel. Netw.*, 2(4):329–342, 1996.

[RRR+99] S. Radhakrishnan, N. Rao, G. Racherla, C. Sekharan, and S. Batsell. Dst - a routing protocol for ad hoc networks using distributed spanning trees. In *IEEE Wireless Communications and Networking Conference*, pages 100–104, 1999.

[SL01] Ivan Stojmenovic and Xu Lin. Loop-free hybrid single-path/flooding routing algorithms with guaranteed delivery for wireless networks. *IEEE Trans. Parallel Distrib. Syst.*, 12(10):1023–1032, 2001.

[Tou91] G. Toussaint. Some unsolved problems on proximity graphs, 1991.

[Urr02] Jorge Urrutia. Routing with guaranteed delivery in geometric and wireless networks. pages 393–406, 2002.