Journal of Computer Science and Cybernetics, V.30, N.4 (2014), 377–396
DOI: 10.15625/1813-9663/30/4/3965

ON HOLE APPROXIMATION ALGORITHMS IN WIRELESS SENSOR NETWORKS

NGUYEN PHI LE, NGUYEN KHANH-VAN

Hanoi University of Science and Technology
lenp@soict.hust.edu.vn; vannk@soict.hust.edu.vn

Abstract. Routing holes in sensor network are regions without operating nodes. They may occur due to several reasons, including cases caused by natural obstacles or disaster suffering areas. Determining the location and shape of holes can help monitor these disaster events (such as volcano, tsunami, etc.) or make smart, early routing decisions for circumventing a hole. However given the energy limit of sensornets, the determination and dissemination of the information about the exact shape of a large hole could be unreasonable. Therefore, there are some techniques to approximate a hole by a simpler shape. In this paper, the authors analyze and compare two existing approximation approaches that are considered as the most suitable for the sensor network, namely the grid-based and the convex-hull-based approaches. And a new algorithm of the grid-based approach is also introduced. The performances of all the mentioned algorithms are under analysis and evaluation in both theoretical and experimental perspectives. The findings show that grid-based approach has advantages in saving network energy and providing a finer image of the hole while the convex hull approach is better for making shorter hole-bypassing route but not much.
Keywords. Wireless sensor networks, routing holes, load balancing, energy efficiency.

1. INTRODUCTION

Wireless sensor networks (WSN) have a wealth of applications. Especially, they are widely used in monitoring and investigating certain landscapes or environments which may be too large or remote for deploying wired network infrastructure or of too harsh conditions that are not suitable for traditional surveillance by human beings. Different from early wired networks, sensor networks can contain a large number of nodes which may be up to hundreds or even thousands. Furthermore, the sensor nodes are only equipped with very limited power source and almost non-rechargeable. These characteristics make it hard to maintain the frequent operations of these network nodes, the life of which can be cut short for demanding workload. The failure of nodes due to energy exhaustion or physical destruction may lead to the occurrence of *holes*, i.e. the regions where the nodes have died out and hence, no longer participated in the network communication. Besides, the holes in sensor networks can also be formed either due to the presence of some geographical obstacles such as buildings, lakes or because of the failure of sensor nodes due to external destroying (e.g. fire, earthquake, etc).

Locating and marking the hole is an well-known problem [22] with two important applications in *environment monitoring* and *geographic routing*. First, sensor networks have been introduced to monitor and control natural disasters. The emergence of a hole usually brings certain information about certain important events in the area such as the occurrence of a disaster or the emergence of a new obstacle. Naturally, sensor networks can be considered useful tools to monitor the landscape of the disaster area, especially the border of the suffering area.

Second, the study of geographical routing is a very active area (a brief review is in section 1.2.) where locating holes is an important issue since the forwarding packets are not possible inside a hole. The hole location is usually determined by the location of the boundary nodes. Once this hole info is determined it can be disseminated to the surrounding area to help improve the routing mechanism. Knowing the presence of a hole in advance can certainly help the nodes to find efficient routes going around the hole. Figure 1 illustrates this using a scenario with a large hole which has a rough face. Fig. 1(a) shows an unnecessarily long route which could be formed as without the awareness about the hole [1]. While fig. 1(b) shows a much shorter route which could be formed by using the hole info (such routes are usually called *escape* or *detour* routes). Thus, knowing about a large hole in advance can help to find shorter route and also to avoid concentrating traffic around the hole boundary (more in section 1.2.).

The hole approximation problem can be seen as a natural extension from the hole locating problem. As sensor networks are usually deployed in large scale, the size of a hole can also be very large. Therefore the determining and/or disseminating the complete information of this hole's boundary could be unaffordable, given the power limit of sensor nodes. More specifically, the above mentioned approach of geographical routing using hole awareness has a crucial drawback. That is, the network lifetime could be significantly reduced because of extra resource consumed by the task of disseminating and storing the information about the hole boundary, the cost of which is directly proportional to the size of data needed to describe this area. Several mechanisms have been proposed to deal with this problem, where the common approach is to approximate the hole by a somehow simpler shape. Thus, the problem of approximating the hole shape can have a significant importance.

Applied in a geographic routing scheme, a good *hole approximation (HA) algorithm* does not only improve the routing process, especially in *sensor networks with large holes*, but also helps to save communication and energy, and thus prolongs the sensor network life. In the opposite, a poor HA algorithm with a large approximation error can lead to longer hole-bypassing routes, i.e. wasting energy for communicating. Figure 1(c) illustrates an example of this. In this example, the hole is approximated by a covering circle which can make the routing path become longer because of the big difference between the hole area and the approximate area (the circle). Thus, the approximate shape (of the approximate area) should be chosen as simple as possible to make compact the description of the *approximate area info*. However, if the approximate shape chosen is too simple and rigid, the difference between the hole and the approximate area could be large, significantly increases the bypassing route length. This trade-off between these network performance factors is the philosophy that influences our analysis framework on HA algorithms.

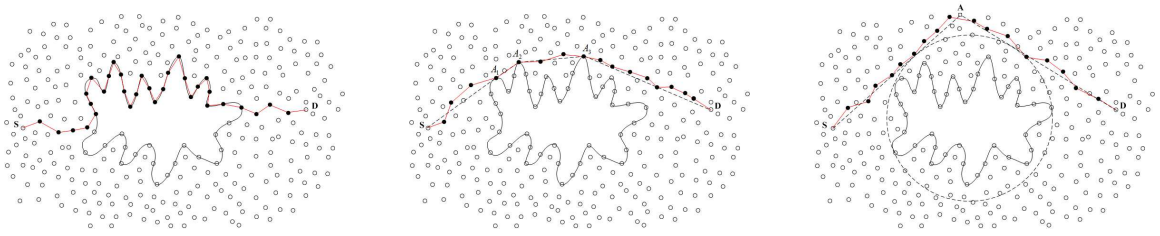
In our opinion, hole approximation is also important in monitoring disaster area. When a disaster has just struck, initially, the border of the suffering area is fast developing; thus, to monitor the size and shape of this area by using sensor network, a HA algorithm must be fast and efficient.

Thus, the hole approximation problem is well motivated and the authors believe that the work can be an useful initial contribution in the study of constructing and analyzing HA algorithms.

1.1. Contribution

In this paper, a study is conducted on this hole approximation problem in an analysis approach where both theoretical and simulation results are provided. To provide a rigorous evaluation framework on

[1] Traditionally, the approach of mixing greedy and perimeter routing is used (e.g. the GPSR protocol in [11]) and hence, the routes can be unnecessarily long if the hole boundary is rough. This also causes heavy traffic concentrating on the hole boundary.



(a) A long route traditionally formed (e.g. GPSR) – without the awareness about the hole

(b) A short route cleverly formed by using the hole info

(c) A not-so-short route, possibly formed due to a poor hole approximation

Figure 1: Comparing possible routing paths, avoiding a large hole

HA algorithms, the different performance factors are used: i) the approximation time, ii) the size in bits of the data used for describing the approximate polygon, ii) the approximation error in area, and iv) the routing path stretch which shows how effective an HA algorithm can be for use in geographic routing. While the first two factors show the general efficiency of a given HA algorithm in term of how much time and energy can be consumed, the following two factors show how suitable the algorithm can be per application, i.e. for monitoring the hole (as a disaster surface) or for making escape routes in geographic routing.

The detailed contributions are:

- Proposal of a new off-line algorithm (opposing to the existing on-line algorithm) for the grid-based approximation approach [15].
- Initial results in comparing the two main approaches, approximating by a convex polygon [23] and by a grid-based one [15]. Most significantly, an upper bound on the ratio of routing stretch of the two is given.
- Simulation results comparing all the three considered algorithms: the convex hull based algorithm, the online grid-based one and the off-line grid-based one (proposed in this paper).

The paper is organized as follows: Section 2 briefly reviews the existing approximation approaches. A new off-line grid-based algorithm is proposed in section 3. Section 4 describes the framework to evaluate and compare hole approximation approaches. Section 5 and 6 shows the results for comparing the grid-based and convex hull based hole approximation algorithms by both theoretical analysis and experiments. The paper is concluded in section 7.

1.2. More related work

Routing hole is a critical issue in geographic routing in wireless networks, an active research area with more than a decade of extensive study. Here, data packets are forwarded based on the positional information of the sensor nodes, assuming that they are aware of their physical locations (e.g. equipped with GPS devices). Early approaches are based on greedy forwarding where a packet is forwarded to the 1-hop neighbor that is closest to the destination. However, this approach can lead into the local minimum phenomenon on the face of a hole (i.e. no neighbor closer to the destination than the current node). To bypass such a hole, traditional proposals appropriately switch between greedy and perimeter forwarding modes, as in the GPSR protocol [11] and several follow-ups [3, 14, 13, 12]. However as many authors have pointed, the traffic concentration on the face of a hole can gradually

degrades the network performance. Subramanian et al. [18] show that GPSR and the likes could cause the network throughput capacity [9] to significantly drop from $\tilde{\Theta}(1/\sqrt{n})$ to just $O(1/n)$ in a scenario where a hole occupies a large part of the network area.

A number of proposals and techniques have been created to deal with the problem of locating network holes [5, 11, 3, 7, 19]. In these, holes are determined by different ways such as ones based on planar graphs [11, 3, 19] or some other geographic approaches [5, 7]. This hole location info then can be used for making escape route to go around the hole efficiently (possibly through disseminating this hole info to the surrounding area). As mentioned above, the size of a hole can be significant large; thus, the determination and/or dissemination of this hole boundary information could be unaffordable given the power limit of sensor nodes. Thus the HA problem naturally emerges.

Although shape approximation of the holes in sensor networks has not been studied yet as an independent problem, a number of approximation-based techniques have been attempted in dealing with holes. These techniques target better mechanisms and algorithms for geographic routing in wireless sensor networks. In an often used simple approach, the hole is approximated by an area which has a rather simpler, common shape such as an ellipse [20], a circle [24, 6, 8, 21] or a hexagon [24, 10], etc. (which is the minimum one in this shape that can fully cover the hole). These common shapes are considered to use as a too simplistic approach, which could results in shortcomings such as large approximation error as well as large routing path stretch (although it could come good in term of short approximation time, and small spreading data size). Therefore the authors do not include this simple shape approach in this effort to fully evaluate and compare the apparently stronger approximation algorithms. The two selected approaches we select to focus on will be discussed closely in section 2..

Finding a convex k-gon enclosing a given n-gon that has the minimum perimeter is a challenging problem in computational geometry which still remains open [1, 2, 4, 17, 16]. In [17], De Pano proposed to compute the minimum perimeter triangle enclosing a given convex polygon, using an $O(n^3)$ algorithm, which was also followed by several improved ones and finally, an linear-time algorithm by Bhattacharya et al. [2] (triangle is also the only case known for linear-time computable). Quite recently, an $O(kn^3/\varepsilon)$ algorithm has been proposed [16] to compute a convex k-gon enclosing n-gon.

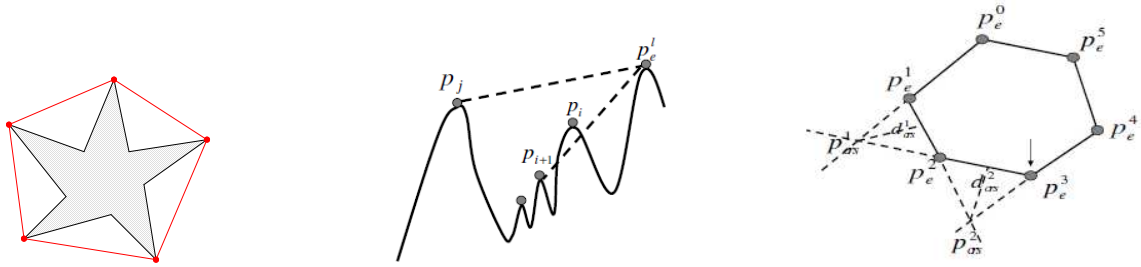
2. HOLE APPROXIMATION APPROACHES

Amongst many such existing approximation-related techniques, *the convex hull* and *the grid-based* approach are the most suitable for the sensor network as their simplicity and efficiency.

In the convex hull approach, the idea is to find a convex hull which can cover the hole. The convex polygon approach (for used in sensor networks) is proposed in [23] where the authors try to improve the routing mechanism based on the visibility graph, originally proposed in [19]. In these papers, holes are considered as obstacles that hinder the visibility between network nodes and thus, using convex polygon for hole approximation is perfectly justified.

In the grid-based approach [15], the main idea is to approximate a hole's boundary with a simpler polygon whose edges are aligned with a given square grid, and thus achieve a grid-based polygon which is easy to describe, convey information and disseminate to the surroundings.

In both above approaches, the size of the approximate polygon (number of vertices) can be requested as a prior condition, and the approximate polygon should be the minimum cover with such a requested size. In a quick comparison, *the grid-based approach tries to closely approximate a hole,* while *the convex polygon approach tries to capture the factor of visibility obstacle that a hole can create for any given pair of source-destination nodes.* Thus, in theory, the grid-based approach would give a finer image of the hole boundary while the convex hull approach would be more efficient



(a) An example of convex hull based A-polygon

(b) Determining the approximate vertex

(c) Vertices reduction

Figure 2: Convex hull based hole approximation algorithm (figure from [23])

for supporting geographic routing. Below, for short notation *A-polygon* is often used in replacement of *approximate polygon*.

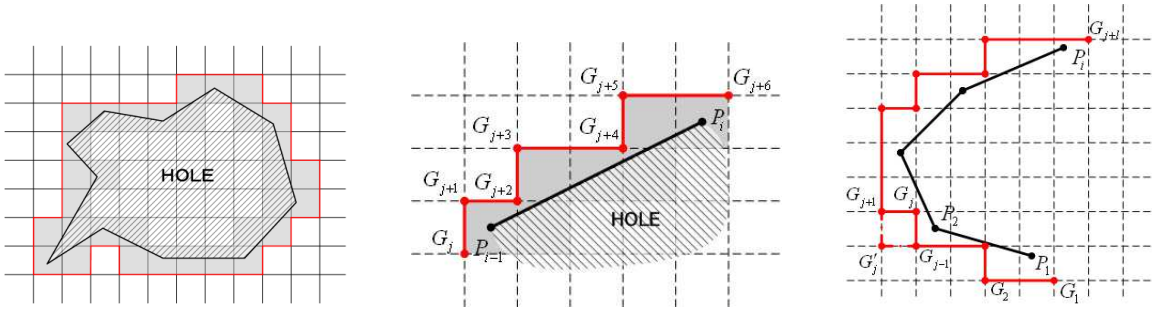
In the following we will briefly introduce these two approaches are briefly introduced. As mentioned above, the two approaches under analysis share the feature of generality in that they allow to compute A-polygons for *the number of the vertices as a prior given value*. They both start with the full hole boundary polygon (computed by using e.g. the BoundHole algorithm in [5]) and then using a proper process to trim off some vertices to meet the prior required vertex number as well as the required shape property (complex vs. grid-based). Both use certain optimization techniques to minimize the precision loss due to this trimming process. By using this feature of limiting the vertex number both approaches gain in the reduction of hole shape information, and thus energy saving in the dissemination phase.

More specifically, these hole approximation schemes contain two processes: *approximation process* and *simplification process*. The former is to approximate the hole by a simpler polygon and the latter is to trim-off the A-polygon so that the number of the vertices does not exceed a predefined threshold. These two processes can be conducted somehow simultaneously. Both approaches also use this common mechanism: a special message, which called the *measuring packet*, is initiated and forwarded along the hole boundary for collecting info on the boundary nodes. Let call it the *M-packet* for short. When the M-packet arrives at any intermediate node, this current node performs two operations belonging to mentioned processes respectively and stores the results (position of the vertices of the A-polygon) to the M-packet.

Above the main concepts and the similarity between these two approaches are discussed. And below, the different techniques being used in these two will be under review.

2.1. Convex hull based approximation

In [23], the authors propose an approach to approximate a given hole by a convex hull. For a given hole, this hole's boundary can be seen as a concave polygon. The convex hull of the hole is a convex polygon which covers this boundary polygon while its vertices are selected from that of the boundary polygon (see Figure 2). In this approach, the hole boundary is determined by a topological method described in [22]. The M-packet is created by an *initiator* node which is selected from the boundary nodes. The M-packet then traverses through the nodes on the hole boundary in the counter-clockwise direction. The convex hull is constructed during this voyage: its list of vertices is to be gradually added and stored into the M-packet.



(a) Approximate polygon (b) Boundary approximation (c) Boundary simplification

Figure 3: Online grid based hole approximation algorithm

When the M-packet arrives at a boundary node, this node performs both *approximation* and *simplification* operations. The approximation operation is to determine if this current node could be included as a vertex of the convex hull. In that case it will be added to the vertex list maintained in the M-packet. Note that, if this vertex list is too long, i.e. the number of the vertices exceeds a predefined threshold, the simplification operation is taken to trim-off some of the vertices, making the vertex size of the convex hull is always under this threshold. Below these two operations are briefly explained. For a full explanation, please see [23].

When M-packet arrives at a given node p_i, this current node determines if it should be included into P_e, the vertex list of the convex hull being constructed (see Figure 2(b)). The key idea here is to check if the convexity property will be still preserved with p_i included ; this could be done by checking if p_i lies on the right side of $p_e^l p_{i+1}$, for some already selected vertex $p_e^l \in P_e$ ($0 \leq l \leq |P_e|$, the size of the current P_e so far). Successful check means that p_i is the farthest visible boundary node from p_e^l so far, i.e. the convexity is still perserved and so, p_i is added to P_e. Later, however, if another boundary node p_j, $j > i$, also visible from p_e^l is found, p_i is deleted from the set P_e.

Figure 2(c) helps to illustrate the algorithm for the simplification. When M-packet reaches node p_i, this node checks if $|P_e|$ is greater than the predefined threshold, in which case the simplification will be performed as briefly reviewed below. For every four consecutive vertices $p_e^{i-1}, p_e^i, p_e^{i+1}, p_e^{i+2}$ in P_e we determine the intersection point P_{icrs}^i between the two lines $p_e^{i-1}p_e^i$ and $p_e^{i+1}p_e^{i+2}$. Amongst the P_{icrs}^i's choose the one with the shortest distance to $p_e^i p_e^{i+1}$ and then use it to replace the corresponding p_e^i and p_e^{i+1} in P_e. This replacement of two existing nodes by one new a clearly reduces the size of P_e by 1.

2.2. Grid-based approximation

In the previous work [15], an approach is introduced to approximate a hole's boundary with a simpler shape, *hole-covering polygon*, which also satisfies the following conditions:

- *Grid alignment:* Its vertices and edges are the nodes and edges of a given square grid (thus, the polygon shape can be easy to manage);
- *Control of approximation error:* Its number of vertices does not exceeds an predefined threshold N where N is a constant parameter which determines how large the approximation error can be.

Figure 3(a) represents an example of a hole and its approximate polygon, or *A-polygon* for short. This approach utilizes the algorithm in [5] to determine the hole boundary. Similarly as above, an initiator node is selected creating the M-packet to traverse around the hole. The authors propose to extend this algorithm to determine the A-polygon. When M-packet arrives at a boundary node, this current node performs *boundary approximation* operation to determine the unit grid squares which intersect the edge connecting the current boundary node and the previous boundary node. This edge is approximated by the broken line connecting the vertices of the intersecting unit grid squares. Figure 3(b) illustrates the algorithm. In this figure, edge $P_{i-1}P_i$ is approximated by a broken line $G_jG_{j+1}...G_{j+6}$ (in red color).

When the number of the vertices of the A-polygon found so far exceeds the threshold, the current boundary node performs the *boundary simplification* operation to trim-off the vertices of A-polygon. Figure 3(c) illustrates this simplification at a node P_i. The key idea is to remove the concave angle that occupies the *smallest area*. More specifically, the current node P_i chooses the concave angle $G_{j-1}G_jG_{j+1}$ (an angle with the interior angle $\geq \pi$) of the A-polygon that has the smallest values of $\overline{G_jG_{j+1}} * \overline{G_{j+1}G_{j+2}}$, and then replaces G_j by $G_j^{'}$ (see Figure 3(c)), the image of G_j through a central symmetry where the center is the midpoint of G_jG_{j+2}. By doing so, the vertex size of the A-polygon is reduced by at least 2. This simplification can be repeated until the size of the A-polygon becomes less than N.

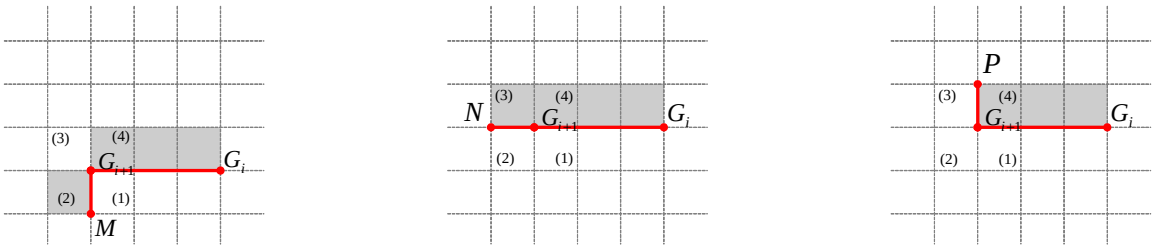
3. OFF-LINE GRID-BASED BOUNDARY APPROXIMATION ALGORITHM

Although the two above mentioned approaches aim to produce different outputs (A-polygons with different shapes) they actually share a lot in principles. First, they are similar in that they both consist of two processes, the approximation process to create an initial polygon which approximates the hole boundary in a specified shaping (convex polygon and grid aligned polygon), and the simplification process to make that initial A-polygon become simpler with fewer vertices (i.e. by trimming-off some vertices). Second, they share the same distributed method in that the two mentioned processes seemingly run simultaneously: the approximation and simplification operations are repeatedly taken when the M-packet visits a new boundary node. This method, that is to simultaneously produce the A-polygon while forwarding the M-packet around the hole, is called as *on-line approximation*.

This online method has the advantage of maintaining the load balance between the boundary nodes. However, the repeating of the simplification operation at many nodes is energy in-efficient. Furthermore, the size of the M-packet may be large and forwarding this packet around causes some extra energy consumption on the boundary nodes.

In the following, the authors propose another grid-based approximation algorithm, which follow an *off-line method*. In this algorithm, the traversing of the M-packet does nothing but collecting the boundary node info. Once this data collection phase finishes, the whole thing of *approximating* and *simplifying* is conducted centrally at only one node - *the initiator*. In other words, the new algorithm consists of two phases: the *data collection*, the M-packet being forwarded around to collect boundary data, and the *boundary approximation*, the initiator computing the A-polygon off-line.

More specifically, in the first phase, the M-packet is forwarded along the hole boundary to determine all unit squares of the grid that intersect the hole boundary. In the second phase, the information collected in the first phase is used to determine the approximate hole which satisfies both two characteristics: its shape is grid aligned and the number of its vertices below a predefined threshold. Below these two phases are described in details.



(a) a(2) = 1 (b) a(2) = 0 and a(3) =1 (c) a(2) = 0 and a(3) = 0

Figure 4: Off-line grid based hole approximation algorithm

In the *data collection* phase, all nodes are found if they are on the boundary of a hole using the algorithm described in [5]. The first node that discovers the hole forms a special packet and forwards it around the hole boundary nodes to determine the nodes with the minimum and maximum *x-coordinate* and *y-coordinate*. After that, the node with the minimum *y-coordinate* (and minimum *x-coordinate* if there are multiple nodes with the same minimum *y-coordinate*), which called the initiator, initiates a packet and forwards it along the hole's boundary to determine the unit square grids which intersecting the hole boundary. This initiator node is denoted as H_0 and the packet as M-packet. This M-packet contains an array *bmp[m][n]*, where $m = \lfloor \frac{x_{max}-x_{min}}{r} \rfloor$, $n = \lfloor \frac{y_{max}-y_{min}}{r} \rfloor$ (x_{min}, y_{min} are the minimum values of *x-coordinate* and *y-coordinate*, x_{max}, y_{max} are the maximum values of *x-coordinate* and *y-coordinate* of the nodes on the hole boundary). Each item $bmp[i][j]$ of array *bmp* represents the unit grid square which has the center with the coordinates of $((i + \frac{1}{2})r, (j+\frac{1}{2})r)$. All items of *bmp* are initiated with the value of *0*. When the M-packet arrives at a boundary node, the current node sets $bmp[i][j]$ to *1* if the corresponding unit grid square of $bmp[i][j]$ intersects the edge connecting the current node and the previous boundary node.

After the M-packet travels around the hole boundary and goes back to the initial node, the initial node conducts the *boundary approximation* phase to determine if the vertices of an unit grid square stay on the boundary of the approximate polygon or not. Fact 1 below shows the way to do that. This fact is quite simple so the proof is omitted.

Fact 1 *Assume G_i and G_{i+1} are two consecutive vertices of the grid that stay on the boundary of the A-polygon in the clockwise order. Denote (1), (2), (3), (4) as the unit squares which has G_{i+1} as an vertex and let M,N,P be the adjacent vertices of G_{i+1} as shown in Figure 4. Let (x_i, y_i) be the coordinates of the center of (i) and a(i) be the value of bmp $\left[\lfloor \frac{x_i}{r} \rfloor\right] \left[\lfloor \frac{y_i}{r} \rfloor\right]$, then:*

- M is an vertex of the A-polygon if and only if a(2)=1 (figure 4(a)).
- N is an vertex of the A-polygon if and only if a(2)=0 and a(3)=1 (figure 4(b)).
- P is an vertex of the A-polygon if and only if a(2)=0 and a(3)=0 (figure 4(c)).

Note that, if the number of the vertices of the A-polygon exceeds the predefined threshold *N*, the initial node performs the *boundary simplification* process described in [15] to trim-off the vertices.

It can be seen that, by using this very short array, info encoded by *0*-and-*1* bits, instead of a list of coordinates of the vertices of the A-polygon, the M-packet size has been reduced much compared to that in the online grid based or the convex hull based algorithms.

4. A FRAMEWORK FOR ANALYSIS AND COMPARISON

So far we have discussed three hole approximation (*HA*) algorithms: the convex hull based, the on-line grid based, and the off-line grid based algorithms (the last one is newly proposed in this paper). It can be seen that each algorithm has its own strengths and weaknesses. The first two algorithms are deployed "on-line" and in a distributed manner. This distributed approach helps to maintain a good load balance between the boundary nodes. However, the repeating of the simplification operation and the possibly large size of the M-packet can impose a heavy task on the boundary nodes. On the other hand, the third algorithm causes a load imbalance due to the heavy load on the initiator node which has a central role. However, this centralized mechanism helps to reduce the M-packet size as well as the simplification workload, which obviously saves energy and resource of the network. Beside, it is not to forget that the two grid-based algorithms can produce A-polygons fitter to the hole than the convex hull based algorithm does. They can even produce this fitness as much as desired by controlling the size of the unit square while this is impossible in the convex hull approach.

The above are only some general evaluation remarks, in the rest of this section the framework for analyzing and comparing *HA* algorithms is described in details. Then in the next sections, this framework will be used to analyze and compare the three above mentioned HA algorithms: by theoretical tools (in section 5) and experiments using simulation (in section 6). The four evaluation criteria which can be defined by the following factors are suggested to use. The generality is concerned so that this framework of criteria can be used not only to compare the existing algorithms but also to evaluate other new HA algorithms and approaches to come by latter.

1. Approximation time: the total time spent by the approximation process when a given HA algorithm is used; i.e. this factor shows how fast a given HA algorithm can be.

The next two factors are used to show how economical in communication a given HA algorithm can be, i.e. indicating how much traffic can be introduced due to a HA process.

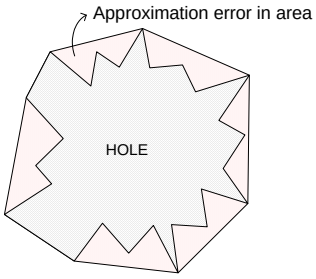
2. Approximate polygon info size (in short, *A-info size*): the size in bits of the data used for capturing the A-polygon description, which is basically the data the M-packet carries when it finishes cycling round the hole and comes back to the initiator. [2] Clearly, this factor indirectly indicates how much information will be communicated during a HA process. Figure 5(a) and 5(b) illustrate a hole and its corresponding convex hull-based A polygon and grid-based A-polygon. In this example, the number of the vertices of the convex hull-based A-polygons and grid-based A-polygon is 8 and 12, respectively. Assumed that we need $8B$ (bytes) in memory to record the coordinates of an vertex, then we need $64B$ to record the convex hull-based A-polygon and $48B$ to record the grid-based A-polygon. [3]

Alternatively, another factor can be used, the *total size of approximation messages*. This is the total size in bits of all the packets which are sent between the boundary nodes in order to collect information of the hole in general (and to determine the A-polygon in two considered approaches). In the two considered approaches, this basically is the sum of the size of all the M-packets being sent between two adjacent boundary nodes.

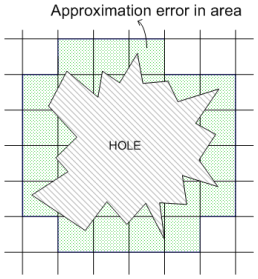
The reason why the two above alternative factors are introduced is while the former (approximate polygon info size) is more specific to the considered approaches and then more suitable for their theoretical analysis, the latter is more general, i.e. can be used for all approaches including ones that

[2] The A-polygon is determined by the location of the vertices, thus this approximate data is simply the list of coordinates of these vertices, or a compressed form of that.

[3] As proved in section 5.1. we do not need to obtain the location of all vertices of the grid-based A-polygon but only a half of it.



(a) Convex hull-based approach



(b) Grid-based approach

Figure 5: Illustration of the approximate error in area

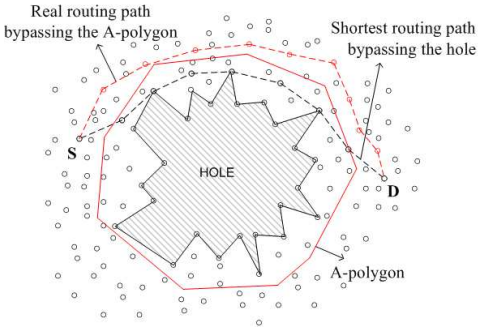


Figure 6: Illustration of the routing path stretch

do not use polygons for hole approximation. However, the latter can only be used in experimental analysis (by simulations).

3. Approximation error in area (in short, *approximate error*): The difference between the area of the A-polygon and the hole. Clearly, this reflects how well an approximation algorithm captures the hole shape and size. Applications that seek reports about disaster surface will require this approximation error to be as small as possible. This approximate error in area is illustrated in figure 5: the red area represents the approximate error of the convex hull-based A-polygon and the green area represents the approximate error of the grid-based A-polygon.

4. Routing path stretch: This factor is particularly defined for the application of HA in geographic routing. As the A-polygon fully covers the hole, the shortest route length should be increased if one try to route around the former rather than the latter. Thus, the routing path stretch is the ratio between the hop-count of the shortest route that bypasses (must avoid to hit) the A-polygon and the counterpart that bypasses the original hole. Figure 6 illustrates the definition of the routing path stretch. In this figure, the red dotted line represents the routing path which bypasses the A-polygon and the black dotted line represents the shortest routing path which bypasses the hole. In this example the routing path stretch is $11/9$.

Clearly, for a given approximation algorithm, while the first two criteria consider the general efficiency of this algorithm in term of how much time and energy can be consumed, the latter two ones show how suitable the algorithm can be per application, i.e. for monitoring the hole (as a disaster surface) or for making escape routes in geographic routing.

5. THEORETICAL ANALYSIS

In this section, the performance of two HA approaches (i.e. convex hull based and grid-based) are under evaluation and comparison using two of five metrics mentioned in the above evaluation framework: approximate polygon info size and routing path stretch. First it is shown that, conditioned on the same number of A-polygon vertices, the A-info size in the grid-based approach is only half of that in the convex hull approach. Second, a small constant upper bound on the ratio between the routing stretches is given by using the convex hull and the grid-based approaches, respectively. Informally, the findings show that using the convex hull approximation approach is not much better than using the grid-based one in finding short route that bypasses the considered hole.

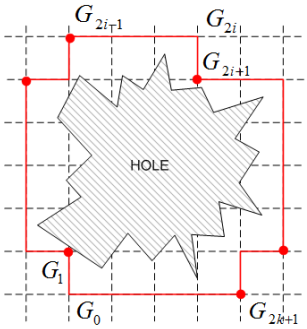


Figure 7: Grid-based A-polygon
◇: the grid-based A-polygon of the hole
●: A-vertices

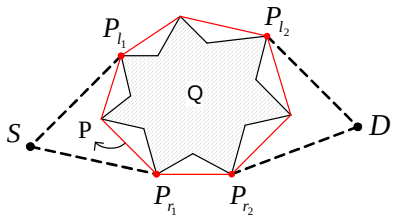


Figure 8: Convex hull, visible vertices and E-shortest path
◇: the convex hull P of polygon Q
●: visible vertices of S and D
The shortest path of (S, D) must be the shorter path between $SP_{l_1}...P_{l_2}D$ and $SP_{r_1}...P_{r_2}D$

5.1. Approximate Polygon Info Size

Consider the A-polygons with n vertices, in the following we will analyze and compare the data size needed to store the information of A-polygon received by grid-based algorithms and convex hull algorithm. As defined in [15], given a considered hole and a natural number n, a *grid-based A-polygon* with n vertices is a *n-gon* that covers the hole and that has been shaped using the grid alignment: its n vertices and edges are the nodes and edges of a given square grid (thus, the polygon shape can be easy to manage). Such a grid-based A-polygon can be obtained using a grid-based hole approximation algorithm such as the one in [15], which works in an on-line distributed regime and tries to minimize approximation error in area. Picture 7 illustrates a grid-based A-polygon. Note that to describe or communicate about the shape of such an A-polygon, it is not needed to obtain the location of all vertices of A-polygon. To save memory, only information about this kind of vertices as defined below is needed to obtain.

Definition 1 *Given a hole H and a square grid, assumed that G is a grid-based A-polygon with the set of vertices be $G_0, G_1, ...G_n$ sorted in the clockwise order. Then, a vertex of G is defined as an A-polygon determining vertex (or A-vertex for short) of G if and only if it satisfies at least one of the following properties:*

- It is the first vertex of an horizontal edge (i.e. parallel to the x-axis)
- Its x-coordinate and y-coordinate differ from that of the preceding A-vertex

Fact 2 *Assumed G is a grid-based A-polygon with the vertex number of n. Then, the coordinates of all vertices of G can be deduced from the coordinates of only A-vertices.*

Fact 2 means that by the nature of a grid, a grid-based A-polygon can be determined by using its A-vertices only, while the number of these vertices is clearly, only a half of the total number of all vertices. Such a thing can not be applied for convex polygons; this implies that, conditioned on a given number (n) of the A-polygon vertices, the grid-based approach can save by 50% in A-info size, compared to the convex hull approach.

Proof. Figure 7 illustrates an example of A-vertices. Here, choose G_1 as the first *A-vertex* which is the first vertex of a horizontal edge; hence, G_3 becomes the next A-vertex since its both coordinates differ from that of G_1. Similarly, $G_5, G_7, ..., G_{2k+1}$ are also A-vertices while $G_0, G_2, ..., G_{2k}$ are

not. It is clear that, the coordinates of $G_0, G_2, ..., G_{2k}$ can be deduced from the coordinates of the mentioned A-vertices as follows:

$$\begin{cases} G_0^x = G_1^x; G_0^y = G_{2k+1}^y \\ G_{2i}^x = G_{2i+1}^x; G_{2i}^y = G_{2i-1}^y \quad (\forall i = \overline{1,k}) \end{cases} \tag{1}$$

(G_i^x, G_i^y denotes the x- and y-coordinate of G_i, respectively).

5.2. Routing Path Stretch

Assumed that the considered network is dense of sensors everywhere apart from the considered hole and by such an idealism, we can model the shortest $s-t$ routing path as the shortest Euclidean line between s and t. In this section, given a large hole, the Euclidean length of the shortest routing path that bypasses a given grid-based A-polygon and the shortest routing path that bypasses a convex hull A-polygon will be compared. Of course, assumed that both these A-polygons are having the same number of vertices (n). Assumed that the considered network is dense of sensors everywhere apart from the considered hole and by such an idealism, we can model the shortest $s-t$ routing path as the shortest Euclidean line between s and t. In this section, given a large hole, the Euclidean length of the shortest routing path that bypasses a given grid-based A-polygon and the shortest routing path that bypasses a convex hull A-polygon will be compared. Of course, assumed that both these A-polygons are having the same number of vertices (n).

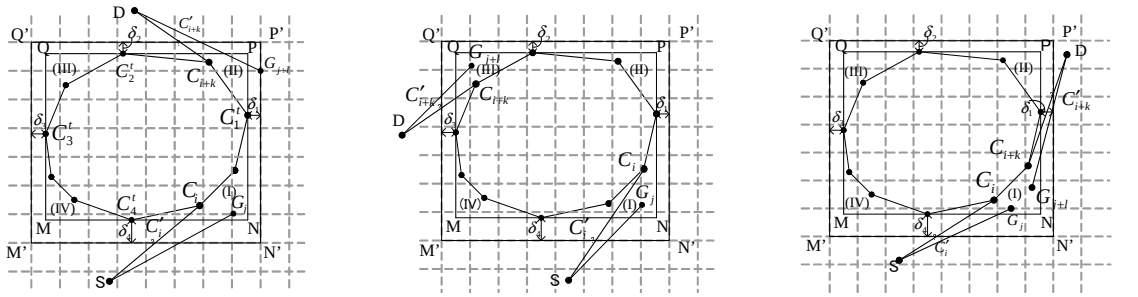
We start by defining the notations and terms used throughout this section. Denote ***H, C, G*** as the hole, the convex hull and the grid-based A-polygon of the hole. Let $C = \{C_1, C_2, ..., C_n\}$ and $G = \{G_1, G_2, ..., G_m\}$ denote the set of the vertices of ***C*** and ***G***, respectively. Assumed, $P_1, P_2, ..., P_i$ are arbitrary points in the plane, then the broken line through $P_1, P_2, ..., P_i$ and its length are denoted by $\overline{P_1P_2...P_i}$ and $\left|\overline{P_1P_2...P_i}\right|$, respectively.

Definition 2 *Assumed that P is a polygon while (S,D) is a pair of source and destination nodes that stay outside of P. Then, the P-bypassing shortest routing path (or P-BSRP for short) of (S,D) is defined as the shortest broken line from S to D which stays outside of P and its length by* $l_P(S, D)$.

Definition 3 *Assumed that P is a polygon while N is an arbitrary node outside P.* P_i*, a vertex of P, is said a visible vertex from N if and only if the line through N and* P_i *does not intersect P.*

Lemma 1 *Assumed that P is a polygon while (S,D) is a pair of source and destination that stay outside of P. Also,* (P_{l_1}, P_{l_2}) *and* (P_{r_1}, P_{r_2}) *are the visible vertices of S and D which lie on the left hand side and right hand side of* $\overrightarrow{SD}$*, respectively. Then, the P-BSRP of (S,D) must be the shorter path between* $S\overrightarrow{P_{l_1}...P_{l_2}}D$ *and* $S\overrightarrow{P_{r_1}...P_{r_2}}D$*, where* $\overrightarrow{P_i...P_j}$ *represents the broken line connecting consecutive convex vertices from* P_i *to* P_j *of P, where the convex vertices of a polygon are the vertices of the convex hull of that polygon (Figure 8).*

Clearly that, $P_{l_1}, P_{l_2}, P_{r_1}, P_{r_2}$ must be the convex vertices of P. Lemma 1 shows that the shortest routing path that bypasses a given hole is the same as the shortest routing path that bypasses the convex hull of that hole, i.e ***H-BSRP*** and ***C-BSRP*** are the same. For grid based A-polygon ***G***, it is clear that ***G-BSRP*** is greater than or equals to ***H-BSRP***. Moreover, ***G-BSRP*** tends to increase when the difference between ***G*** and ***H*** increases. Notice that, this difference tends to be increased when increasing the size of the unit square of the grid.



(a) Case 1: C_i and C_{i+k} belong to two consecutive *C-vertices subsets*

(b) Case 2: C_i and C_{i+k} belongs to two non-consecutive *C-vertices subsets*

(c) Case 3: C_i and C_{i+k} belongs to the same *C-vertices subset*

Figure 9: Proof of Theorem 1

In the following, the relation between ***G-BSRP***, ***C-BSRP*** and the size of the unit square are under analysis. Denote ***MNPQ*** and $M'N'P'Q'$ as the smallest rectangles covering ***C*** and ***G*** respectively, figure 9 results in the following theorem.

Theorem 1 *For any source-destination pair (S,D) staying outside G and C, we have:*

$$l_G(S,D) \leq \sqrt{2} l_C(S,D) + \delta \tag{2}$$

Where δ is the difference between the perimeter of MNPQ *and $M'N'P'Q'$.*

While lemma 1 shows that a convex hull is the most efficient shape (to approximate a hole) for optimizing the routing stretch, theorem 1 shows that, conditioned on the same number of vertices, *a grid-based A-polygon may be a bit worse but still comparable to a convex hull A-polygon in term of the shortest routing path length*. More specifically, the shortest routing path in the first is within multiplication factor $\sqrt{2}$ from that in the second.

Proof. Let $\delta_1, \delta_2, \delta_3, \delta_4$ be the distance between the edges of ***MNPQ*** and $M'N'P'Q'$, then $\delta = 2(\delta_1 + \delta_2 + \delta_3 + \delta_4)$. Let $C_1^t, C_2^t, C_3^t, C_4^t$ be the tangential points of ***C*** to ***MNPQ***. Divide the vertices of ***C*** into four subsets (I), (II), (III), (IV) as shown in Figure 9(a), we call these subsets ***C-vertices subsets***. Assumed $S\overrightarrow{C_iC_{i+1}...C_{i+k}}D$ is ***C-BSRP*** of ***(S,D)***, there are three cases which can occur:

- Case 1: C_i and C_{i+k} belong to two consecutive ***C-vertices subsets***. For example, C_i belongs to (I) and C_{i+k} to (II) (Figure 9(a)).
- Case 2: C_i and C_{i+k} belongs to two non-consecutive ***C-vertices subsets***. For example, C_i belongs to (I) and C_{i+k} to (III) (Figure 9(b)).
- Case 3: C_i and C_{i+k} belongs to the same ***C-vertices subset***. For example, C_i and C_{i+k} are both belong to (I) (Figure 9(c)).

In the following, the theorem in case 1 will be proved, when C_i and C_{i+k} belong to two consecutive ***C-vertices subsets***. Similar results for the other cases can be deduced from this case.

Without loss of generality, assumed that C_i belongs to (I) and C_{i+k} to (II) (Figure 9(a)). As $S\overrightarrow{C_iC_{i+1}...C_{i+k}}D$ is ***C-BSRP*** of (S,D), $C_1, C_2, ..., C_n$ must stay on the left side of $\overrightarrow{SC_i}$ and right side of $\overrightarrow{DC_{i+k}}$. Assumed G_j, G_{j+1} are visible vertices of ***S,D*** which stay on the right side of $\overrightarrow{SD}$

then G_j must stay on the right side of $\overrightarrow{SC_i}$ and G_{j+l} must stay on the left side of $\overrightarrow{DC_{i+k}}$. According to lemma 1, $l_G(S,D) \leq \left|\overrightarrow{SG_j...G_{j+l}D}\right|$. Let ${C_i}', {C_{i+k}}'$ denote the intersections of SC_i and SC_{i+k} with $M'N'$ and $P'Q'$, respectively yields:

$$l_G(S,D) \leq \left|\overrightarrow{SG_j...G_{j+l}D}\right| \leq \left|\overrightarrow{SN'P'D}\right| \tag{3}$$

Using triangular inequation results in:

$$SN' \leq SC_i' + C_i'N'; DP' \leq DC_{i+k}' + C_{i+k}'P' \tag{4}$$

From (3) and (4), we have:

$$l_G(S,D) \leq \left(\left|SC_i'\right| + \left|DC_{i+k}'\right|\right) + \left|\overrightarrow{C_i'N'P'C_{i+k}'}\right| \tag{5}$$

Denote $(x_i, y_i), ..., (x_{i+k}, y_{i+k})$ as the coordinates of $C_i, ..., C_{i+k}$ and (x_i', y_i'), (x_{i+k}', y_{i+k}') as the coordinates of C_i', C_{i+k}'. Notice that,

$$\begin{aligned}\left|\overrightarrow{C_i'N'P'C_{i+k}'}\right| &= \left|x_i - x_i'\right| + \left|y_i - y_i'\right| + \left|x_{i+k} - x_{i+k}'\right| + \left|y_{i+k} - y_{i+k}'\right| \\ &\quad + \sum_{u=i}^{i+k-1} \left(|x_{u+1} - x_u| + |y_{u+1} - y_u|\right) + 2\delta_1 \\ \Rightarrow \left|\overrightarrow{C_i'N'P'C_{i+k}'}\right| &\leq \sqrt{2\left(|x_i - x_i'|^2 + |y_i - y_i'|^2\right)} + \sqrt{2\left(|x_{i+k} - x_{i+k}'|^2 + |y_{i+k} - y_{i+k}'|^2\right)} \\ &\quad + \sum_{u=i}^{i+k-1} \sqrt{2\left(|x_{u+1} - x_u|^2 + |y_{u+1} - y_u|^2\right)} + 2\delta_1 \\ \Rightarrow \left|\overrightarrow{C_i'N'P'C_{i+k}'}\right| &\leq \sqrt{2}\left|C_i'\overrightarrow{C_i...C_{i+k}}C_{i+k}'\right| + 2\delta_1\end{aligned} \tag{6, 7}$$

From (5) and (7), we can deduce that:

$$\begin{aligned} l_G(S,D) &\leq \sqrt{2}\left|\overrightarrow{SC_iC_i + kD}\right| + 2\delta_1 \\ &\leq \sqrt{2} l_C(S,D) + \delta. \end{aligned} \tag{8}$$

The theorem is proved in the first case. The proof for other cases can be deduced as below:

- Case 2: If C_i and C_{i+k} belong to two non-consecutive ***C-vertices subsets***. For example, C_i belongs to ***(I)*** and C_{i+k} to ***(III)*** (Figure 9(b)):

$$\begin{aligned} l_G(S,D) &\leq \sqrt{2} l_C(S,D) + 2\left(\delta_1 + \delta_2\right) \\ &\leq \sqrt{2} l_C(S,D) + \delta \end{aligned} \tag{9}$$

- Case 3: C_i and C_{i+k} belong to the same ***C-vertices subset***. For example, C_i and C_{i+k} are both belong to ***(I)*** (Figure 9(c)):

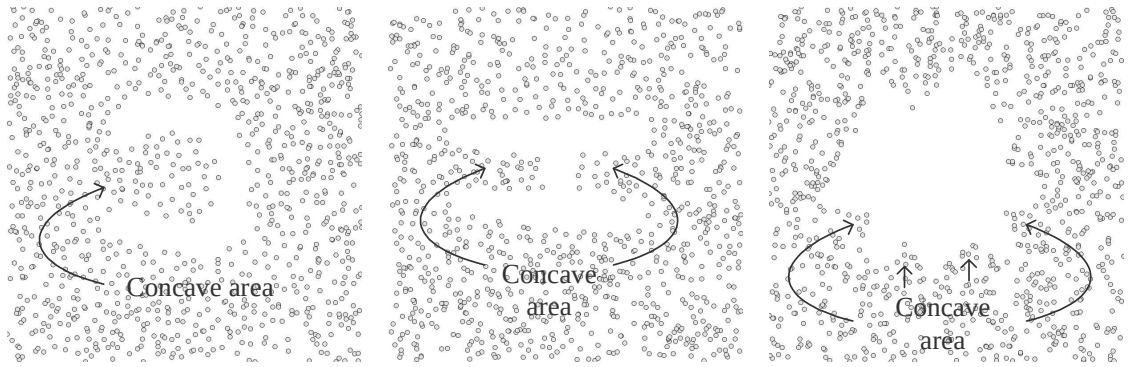
$$l_G(S,D) \leq \sqrt{2} l_C(S,D) \tag{10}$$

From (8), (9) and (10), the theorem is proved.

Furthermore, denote r as the edge length of the unit square of the grid, then $\delta < 8r$ and thus $l_G(S,D) \leq \sqrt{2} l_C(S,D) + 8r$.

6. EXPERIMENTAL EVALUATION

6.1. Simulation setup



(a) C-shape (b) I-shape (c) Star-shape

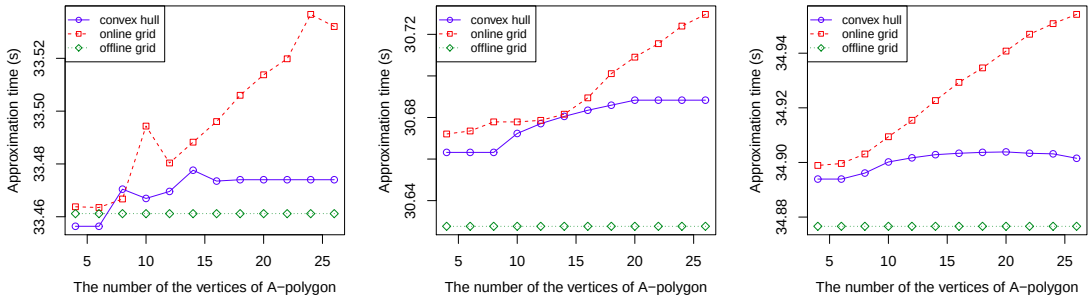
Figure 10: Simulation topologies

In this section, the approximation algorithms using experimental work are under evaluation and comparison. The experiments on the NS-2 simulator are conducted and 802.11 selected as our MAC protocol. Table 1 summarizes the parameters used. In the simulation, 1000 sensor nodes are deployed randomly in an area of $500m$ x $500m$. The holes with many kinds of shape are created randomly. The number of the vertices of the holes is from 50 to 55. Figure 10 represents the network topologies used in the simulation. As shown in this figure, there are three network topologies are used. The first one has a hole with the shape similar to letter *C*. The second one has a hole with the shape like letter *I* and the last one has a hole with the star shape. It is seen that, the hole in the first and the second topologies has a large concave area while the hole in the last one is nearly a convex polygon. With each topology, the authors conduct three hole approximation algorithms and evaluate the performance based on the following metrics:

Table 1: Simulation parameters

Variables	*Values*
Communication range	40m
Number of nodes	1000
Initial energy of each node	1000J
Reception Power	1.0 J/s
Transmission Power	2.0 J/s
Idle Power	0.2 J/s
Sleep Power	0.001 J/s

- *Approximation time:* the time to approximate the hole by the A-polygon.
- *Approximation error in area:* the difference in area between the A-polygon and the hole.
- *E-stretch of the routing path:* The ratio between the Euclidean length of the shortest routing path bypassing the grid-based A-polygons and the shortest routing path bypassing the convex hull.
- *Size of approximation messages:* The total size of the messages which are forwarded to collect information of the hole and determine the A-polygon. This factor reflects the communication overhead introduced by the approximate process.



(a) C-shape (b) I-shape (c) Star-shape

Figure 11: Approximate time

6.2. Simulation results

In the following, G-offline, G-online are used to denote the online grid based algorithm and offline grid based algorithm, respectively.

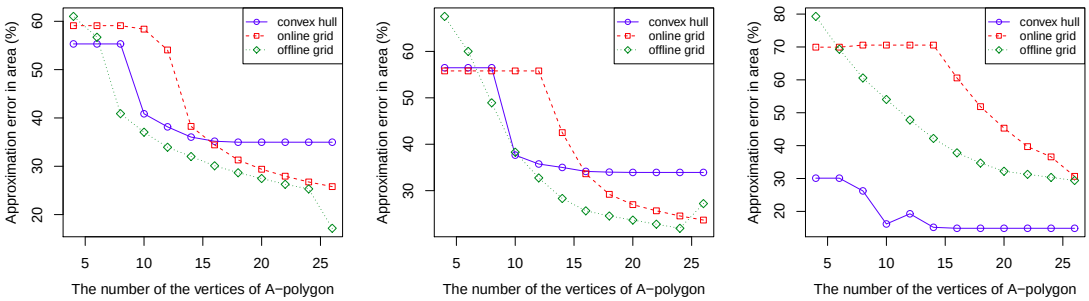
6.2.1. Approximation time

Figure 11 shows the approximation time. The x-axis represents the number of the vertices of the A-polygon and the y-axis represents the approximation time of the three algorithms. It can be seen that, the approximation time of the G-offline is the smallest. The reason is that in the G-offline, the simplification process is performed only one time at the initial node while in the others, this process is performed at every nodes where the number of the vertices of the A-polygon exceeds the threshold. Furthermore, as the M-packets (i.e. the packets which are forwarded to collect information of the hole and determine the A-polygon) in G-offline contains only a binary string which has the size much smaller than the size of the M-packets containing the coordinates of all the approximate vertices in the other algorithms, the transmission time of the M-packets in G-offline is thus much smaller than that in the others.

Moreover, it can be seen that, while the approximation time of the G-online and the convex hull algorithm tends to increase when the number of the vertices of A-polygon increases, the approximation time of the G-offline does not. This is because when the number of the vertices of A-polygon increases, the size of the M-packets in G-online and convex hull based algorithm increases. The increase of the size of approximation messages enlarges the time to transmit the M-packets and thus increases the approximation time. While in the G-offline, the size of M-packets is very small and does not depend on the number of the vertices of A-polygon.

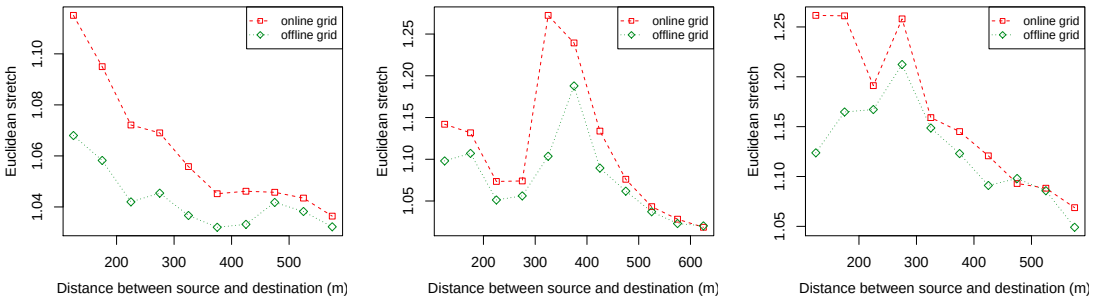
6.2.2. Approximation error in area

The approximate error between the area of the hole and the area of the A-polygons is illustrated in Figure 12. The error is calculated as $\frac{S_{appr}-S_{hole}}{S_{hole}}$, where S_{appr} and S_{hole} are the area of the A-polygon and the hole, respectively. It can be seen that, the approximate error decreases with the increasing of the number of vertices of the A-polygons. However, the approximate error of the grid based A-polygons tends to decrease very fast with the increasing of the number of vertices while the approximate error of convex hull based A-polygon does not. The approximate error of the grid based



(a) C-shape (b) I-shape (c) Star-shape

Figure 12: Approximate error in area



(a) C-shape (b) I-shape (c) Star-shape

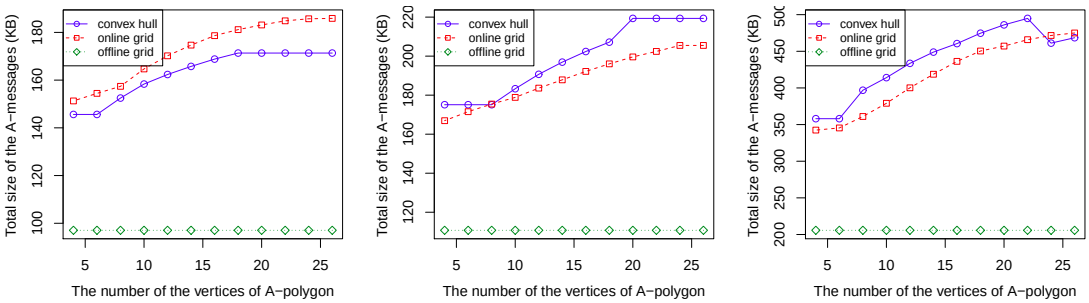
Figure 13: Routing path stretch

A-polygons can be made as small as desired by increasing the number of vertices of the A-polygon while the approximate error of the convex hull asymptotes to a stable value.

As shown in figure 10, the hole with the shape like letter ***C*** and letter ***I*** have a large concave area while the star shape hole is nearly convex (i.e. it has a very small concave area). For a network topology which has the hole with a large concave area, the approximation error of the grid based A-polygon tends to be much smaller than that of the convex based A-polygon. However, for networks with the *nearly convex* hole, the approximation error of the convex hull based A-polygon is the smallest.

6.2.3. E-stretch of the routing path

Figure 13 compares the Euclidean stretch (or *E-stretch* for short) of the routing path when bypassing the A-polygons: using the G-online versus using the G-offline. For an arbitrary pair of source-destination (S,D), denote $l_G(S,D)$, $l_C(S,D)$ as the Euclidean length of the shortest routing path which bypasses the grid-based A-polygon and the shortest routing path which bypasses the convex hull of the hole, respectively. Then, the E-stretch is computed as $\frac{l_G(S,D)}{l_C(S,D}$. In figure 13, the green line represents the E-stretch of the shortest routing path which bypasses the online grid-based A-polygon (or E-stretch of G-online for short) and the red line represents the E-stretch of the shortest routing path which bypasses the offline grid-based A-polygon (or E-stretch of G-offline for short). The *x*-



(a) C-shape (b) I-shape (c) Star-shape

Figure 14: Total size of approximation messages

axis represents the distance between the source and the destination while the y-axis represents the E-stretch. It is clear that the E-stretches are always greater than *1* but it does not exceed *1.3*. This means that the convex hull based A-polygon can achieve a shorter routing path than the grid-based A-polygons but the difference between them is not large. It can be seen that the E-stretch of the G-offline is smaller than that of G-online. This is because the approximation error in area between the A-polygon in G-online and the hole is smaller than that between the A-polygon in G-offline and the hole. However, the difference between these stretches decreases with the increasing of the distance between the source and the destination and when the distance is greater than $500m$, the difference becomes negligible.

6.2.4. Total size of approximation messages

Figure 14 shows the total size of the approximation messages. It is clear that, the size in G-offline is very small compared to that in G-online and convex hull based algorithm. In the best case, G-offline can save more than 60% size of the approximation messages and even in the worse case, G-offline can save more than 40%. Moreover, we can see that the size of the approximation messages in G-offline does not depend on the number of the vertices of A-polygon while the size of the other two algorithms increases rapidly with increasing of the number of the vertices of A-polygon.

7. CONCLUSION AND FUTURE WORK

In this paper, the off-line grid based hole approximation algorithm is proposed. The authors also analyze and compare the performance of three hole approximation algorithms: the online grid based, offline grid based and convex hull based. The theoretical analysis results prove that, with the same number of the vertices, the data of grid-based approximate polygons is only a half of that of convex hull based approximate polygon. The theoretical analysis also shows that, the length of the routing path that bypasses the grid-based A-polygons is not worse than $\sqrt{2}$ times of that by bypasses the convex hull. The simulation results show that the approximation time of the offline grid based algorithm is the smallest while that of the online grid based algorithm is the largest; for the hole which has a large concave area, the approximate error in area between the grid based approximate polygons and the hole is smaller than that of the convex hull; the routing path when bypassing the grid based approximate polygons is longer than but it does not exceed 1.3 times of the routing path when bypassing the convex hull based approximate polygon.

In the future, the authors will consider the relation between the size of the unit square grid and the approximate error of the grid-based A-polygon and the hole. We also evaluate the affect of the three approximate algorithms on different routing algorithms.

REFERENCES

[1] A. Aggarwal and J. K. Park, "Notes on searching in multidimensional monotone arrays (preliminary version)," in *FOCS*. IEEE Computer Society, 1988, pp. 497–512.

[2] B. K. Bhattacharya and A. Mukhopadhyay, "On the minimum perimeter triangle enclosing a convex polygon." ser. Lecture Notes in Computer Science, J. Akiyama and M. Kano, Eds. Springer, 2002, pp. 84–96.

[3] P. Bose, P. Morin, I. Stojmenovir, and J. Urrutia, "Routing with guaranteed delivery in ad hoc wireless networks," in *5th IEEE International Conference*, 2008, pp. 347–352.

[4] J. E. Boyce, D. P. Dobkin, R. L. S. D. III, and L. J. Guibas, "Finding extremal polygons," in *STOC*. ACM, 1982, pp. 282–289.

[5] Q. Fang, J. Gao, and L. J. Guibas, "Locating and Bypassing Routing Holes in Sensor Networks," in *Proc. of INFOCOM'04*, 2004.

[6] F.Li, B.Zhang, and J.Zheng, "Geographic hole-bypassing forwarding protocol for wireless sensor networks," *Communications, IET*, vol. 5, 2011, pp. 737–744.

[7] S. Funke, "Topological Hole Detection in Wireless Sensor Networks and Its Applications," in *DIALM-POMC '05 Proceedings of the 2005 joint workshop on Foundations of mobile computing*, 2005.

[8] F.Yu, S.Park, E.Lee, and S.H.Kim, "Hole modeling and detour scheme for geographic routing in wireless sensor networks," *Journal of communication and networks*, vol. 11, 2009, pp. 327–336.

[9] P. Gupta and P. R. Kumar, "The capacity of wireless networks," *IEEE Transactions on Information Theory*, vol. 46, 2000, no. 2, pp. 388–404.

[10] H.Choo, M.Choi, M.Shon, and D.S.Kim, "Efficient hole bypass routing scheme using observer packets for geographic routing in wireless sensor networks," *ACM SIGAPP Applied Computing Review*, vol. 11, 2009, pp. 7–16.

[11] B. Karp and H. T. Kung, "GPSR: Greedy Perimeter Stateless Routing for Wireless Networks," in *Proc. of MOBICOM'00*, 2000, pp. 243–254.

[12] F. Kung *et al.*, "Geometric Ad-hoc Routing: Of Theory and Practice," in *Proc. of ACM PODC*, 2003.

[13] F. Kung, R.Wattenhofer, and A. Zollinger, "Worst Case Optimal and Average-Case Efficient Geometric Ad-hoc Routing," in *Proc. of ACM MobiHoc*, 2003.

[14] F. Kung, R. Wattenhofer, and A. Zollinger, "Asymptotically Optimal Geometric Mobile Ad-hoc Routing," in *Dial-M*, 2002.

[15] N. P. Le, B. T. Quan, N. T. Hieu, and N. K. Van, "Efficient approximation of routing holes in wireless sensor networks," in *ACM Proc. of the Second Symposium on Information and Communication Technology*, 2011.

[16] J. S. B. Mitchell and V. Polishchuk, "Minimum-perimeter enclosures." *Inf. Process. Lett.*, vol. 107, 2008, no. 3-4, pp. 120–124.

[17] P. N.A.A De, "Polygon approximation with optimized polygonal enclosures: applications and algorithms." *Communications, IET*, 1987.

[18] S. Subramatian, S. Shakkottai, and P. Gupta, "On Optimal Geographical Routing in Wireless Networks with Holes and Non-Uniform Traffic," in *Proc. of IEEE INFOCOM*, 2007.

[19] G. Tan, M. Bertier, and A.-M. Kermarrec, "Convex partition of sensor networks and its use in virtual coordinate geographic routing." in *INFOCOM*, 2009, pp. 1746–1754.

[20] Y. Tian *et al.*, "Energy-Efficient Data Dissemination Protocol for Detouring Routing Holes in Wireless Sensor Networks," in *Proc. of IEEE Intl. Conf. on Communications, ICC'08*, 2008, pp. 2322–2326.

[21] G. Trajcevski, F. Zhou, R. Tamassia, B. Avci, P. Scheuermann, and A. A. Khokhar, "Bypassing holes in sensor networks: Load-balance vs. latency," in *GLOBECOM*, 2011, pp. 1–5.

[22] Y. Wang, J. Gao, and J. S. B. Mitchell, "Boundary recognition in sensor networks by topological methods." in *MOBICOM*, 2006, pp. 122–133.

[23] M. Won, R. Stoleru, and H. Wu, "Geographic routing with constant stretch in large scale sensor networks with holes," in *Proc. of WiMob*, 2011, pp. 80–88.

[24] F. Yu *et al.*, "Efficient Hole Detour Scheme for Geographic Routing in Wireless Sensor Networks," in *Proc. of the 67th IEEE Vehicular Technology Conference, VTC'08*, 2008, pp. 153–157.

Received on April 28 - 2014
Revised on October 03 - 2014