JOHN WILEY & SONS, LTD., THE ATRIUM, SOUTHERN GATE, CHICHESTER P019 8SQ, UK

*** PROOF OF YOUR ARTICLE ATTACHED, PLEASE READ CAREFULLY ***

After receipt of your corrections your article will be published initially within the online version of the journal.

PLEASE NOTE THAT THE PROMPT RETURN OF YOUR PROOF CORRECTIONS WILL ENSURE THAT THERE ARE NO UNNECESSARY DELAYS IN THE PUBLICATION OF YOUR ARTICLE

☐ READ PROOFS CAREFULLY

ONCE PUBLISHED ONLINE OR IN PRINT IT IS NOT POSSIBLE TO MAKE ANY FURTHER CORRECTIONS TO YOUR ARTICLE

- This will be your only chance to correct your proof
- Please note that the volume and page numbers shown on the proofs are for position only

☐ ANSWER ALL QUERIES ON PROOFS (Queries are attached as the last page of your proof.)

- Please annotate this file electronically and return by email to the production contact as detailed in the covering email. Guidelines on using the electronic annotation tools can be found at the end of the proof. If you are unable to correct your proof using electronic annotation, please list all corrections and send back via email to the address in the covering email, or mark all corrections directly on the proofs and send the scanned copy via email. Please do not send corrections by fax or post.

Acrobat Reader & Acrobat Professional

- You will only be able to annotate the file using Acrobat Reader 8.0 or above and Acrobat Professional. Acrobat Reader can be downloaded free of charge at the following address: http://www.adobe.com/products/acrobat/readstep2.html

☐ CHECK FIGURES AND TABLES CAREFULLY

- Check sizes, numbering, and orientation of figures
- All images in the PDF are downsampled (reduced to lower resolution and file size) to facilitate Internet delivery. These images will appear at higher resolution and sharpness in the printed article
- Review figure legends to ensure that they are complete
- Check all tables. Review layout, titles, and footnotes

☐ COMPLETE COPYRIGHT TRANSFER AGREEMENT (CTA) if you have not already signed one

- Please send a scanned signed copy with your proofs by e-mail. **Your article cannot be published unless we have received the signed CTA**

☐ OFFPRINTS

- Free access to the final PDF offprint of your article will be available via Author Services only. Please therefore sign up for Author Services if you would like to access your article PDF offprint and enjoy the many other benefits the service offers.

Additional reprint and journal issue purchases

- Should you wish to purchase additional copies of your article, please click on the link and follow the instructions provided: http://offprint.cosprinters.com/cos/bw/
- Corresponding authors are invited to inform their co-authors of the reprint options available.
- Please note that regardless of the form in which they are acquired, reprints should not be resold, nor further disseminated in electronic or print form, nor deployed in part or in whole in any marketing, promotional or educational contexts without authorization from Wiley. Permissions requests should be directed to mailto: permissionsuk@wiley.com

SECURITY AND COMMUNICATION NETWORKS
Security Comm. Networks 2010; **3**:1–10
Published online in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/sec.271

RESEARCH ARTICLE

Local geometric algorithm for hole boundary detection in sensor networks

Amit Shirsat[1]* and Bharat Bhargava[2]

[1] US Office: 2GA 4358, 4401 Great America Parkway, Santa Clara, CA 95054, USA
[2] US Office: LWSN 2116F, 305 N. University Street, West Lafayette, IN 47907, USA

ABSTRACT

We present a hole boundary detection algorithm for sensor network which identifies the geographical boundary of voids in the network assuming the relative geographic information of only 2-hop neighbors. We formalize the fuzzy notion of a hole by relating it to the graph theoretic concept of a chordless cycle enclosing a face in an arrangement of lines. Furthermore, the algorithm for detecting such topological holes is distributed, O(k) per node computation (for k 2-hop neighbors) and requires synchronization between nodes that are no more than 2-hops away. The algorithm takes a local best-effort approach and does not verify if the nodes indeed form a closed polygonal loop. We believe that this local effort is sufficient based on theoretical observations and experiments. In the simulation tests we conducted, this was hardly a restriction in correctly identifying the simple, closed polygonal loops. We discuss the security implications of the hole detection framework in the context of sensor networks. Copyright © 2010 John Wiley & Sons, Ltd.

KEYWORDS

geometric algorithm; hole detection; security; sensor networks

***Correspondence**

Amit Shirsat, US Office: 2GA 4358, 4401 Great America Parkway, Santa Clara, CA 95054, USA.
E-mail: shirsat@alumni.purdue.edu

1. INTRODUCTION

1.1. Hole boundary detection problem

Ad hoc sensor networks find applications in environments which require little external control over network configuration. The ad hoc networks distinguish themselves from other networks in that they are self-configurable to a significant extent. Automated network configuration is a fundamental task for ad hoc networks. Sensor nodes may have to be deployed randomly in a region during a crisis situation. Such a distribution may create sub-regions with sparse density of nodes in the network. The network level routing protocols have to adapt to the distribution and the energy-levels of the sensor nodes. In order to realize the goal of environmental awareness, we envision that the sensor network be also topology aware. This means that the network by itself can figure out sub-regions which are deficient in sensors or the regions in the sensor map which have low energy content or even terrains which cannot be monitored due to geographical constraints or obstructing objects. One or more of these prevailing conditions creates voids or sub-regions which are devoid of sensors. Higher-level applications or protocols can then determine a strategy to either reinforce such voids with new sensors or discover alternate routes to avoid such voids. In this paper, we discuss and develop the formal definitions required to represent such voids. The topology-awareness problem must not conflict with the goal of energy-awareness, that is, the network must not spend too much energy on creating and maintaining the overlay. Therefore, any solution for topology-awareness must be designed with the view to keep the overheads of communication to a minimum. Communication is the most energy intensive operation for sensor networks, and the energy consumption varies between the second and fifth power of the transmission distance between the nodes [1]. This property drives the design principle of distributed algorithms to be predominantly local; the nodes become aware of the network state using more nodes near-by rather than far-off. Furthermore, this is even more so crucial for networks when topology changes substantially with respect to time either due to mobility, or link failures or node failures. For a targeted measure of efficiency, localized algorithms are designed with a view that any arbitrary node keeps track of at most $O(\log^d n)$ nodes where there are n nodes in a fixed d-dimensional space—the distribution of these tracked nodes decreases exponentially from the tracking node.

UNCORRECTED PROOFS

Copyright © 2010 John Wiley & Sons, Ltd.

In this paper, we present a O(k) per node (where k is the number of 2-hop neighbors) hole boundary detection protocol, which identifies the nodes on a hole boundary assuming the geographic information of only 2-hop neighbors. We formalize the notion of a hole by relating it to the graph theoretic concept of a chordless cycle. Furthermore, the algorithm is distributed and requires synchronization between nodes that are no more than 2-hops away. The algorithm takes a local best-effort approach and does not verify if the nodes indeed form a closed polygonal loop. In the simulation tests we conducted, this was hardly a restriction in correctly identifying the simple (non-intersecting), closed polygonal loops.

Since, topology-aware ad hoc sensor networks create a virtual network overlay on top of the physical distribution, we envisage the hole detection framework to be stacked in between the MAC and the network layers in the OSI protocol stack. In the rest of this section we motivate the discussion on the hole detection problem in the context of applications.

1.2. Motivation

In the scenario discussed in this paper we assume that, the sensor nodes reconfigure themselves into a network topology using a completely decentralized ad hoc mechanism. The sensor nodes are *self-aware*—they discover their neighboring network topology by exchanging beacons or *heartbeat* messages between nodes within their communication ranges. Power optimal connectivity and coverage are some of the challenges to self-configuration discussed in literature [2]. With power optimal connectivity, we require that the network graph is connected and typically has a large component which is k-connected for $1 < k \leq 3$. The range on k ensures that the network is resilient to node failures and congestion. At the same time, we do not require the graph to be dense, because this would lead to more nodes being active routers in the network causing hop delays and interference problems. The coverage problem in sensor networks refers to how much fraction of a region is sensed by the sensors. The detection of voids or *holes* precisely addresses this problem. The simple, closed polygonal region is a representation of the topological boundary of a void in the network graph. The holes represent the closed regions which are possibly under-populated with sensors and are left uncovered. Thus, hole detection is a crucial problem which needs to be addressed to ensure coverage while self-configuring the network topology.

Hole detection can play an important role in greedy stateless geographic routing. In these schemes, a node forwards a packet to one of its neighbors (based on an optimization criteria such as closest neighbor to the destination) without maintaining a partial routing table of intermediate nodes to the destination. A few examples of such stateless protocols are GFG [3], GPSR [4], and compass routing [5]. In the stateless protocols each forwarding node keeps track of at most $O(1)$ nodes in its neighborhood. All greedy forwarding protocols suffer from the *local minimum phenomenon*. This happens when a packet gets stuck on a node which is a local minimum w.r.t. the greeding forwarding criteria. The stuck nodes are part of the hole boundary in the network. If the network is aware of hole boundaries, it can route packets around the holes until the packet gets out of the local minimum position.

The topological boundary detection can be used for storing Geographic Hash Tables (GHTs) [6]. In geographic hashing [7], the hash of a node identifier $\mathcal{I}$ is mapped to a geographic position, represented by its Cartesian coordinates (x, y). Different geographic hashing protocols map this geographical position to a node which is typically close to (x, y). Few examples of such protocols are GLS [8], GLHS [7] for geographic location. If the position (x, y) is enclosed inside a hole boundary, then it could be possible that there is no node close enough to that position, in which case it is hashed to partial subset of nodes on the boundary of the hole. A query packet for the node $\mathcal{I}$ would hit the hole boundary and by traversing the hole boundary, it would eventually reach a node that stores the geographic information for $\mathcal{I}$.

The topological boundary detection can also be used to detect boundaries between secure and insecure nodes, if the insecure nodes are localized and confined to a closed region. Since, the hole detection framework is topological, the definition of an edge can be extended to be a secured connection between two secure nodes which are geometrically proximal. The entire argument for hole boundary detection will still fall through with this renewed definition. The hole boundary will then correspond to the boundary between secure and insecure nodes (and voids) in a network.

The organization of this paper is as follows. We present a brief overview of the existing work in Section 2. We describe the algorithm for hole detection in Section 4. ~~We follow it up in Section 3.2 with some geometrical proofs to bring in some insights into the workings of the algorithm.~~ In Section 5, we remark on the quality of the algorithms by validating through simulations. Finally in Section 7, we conclude with directions for future work and applications.

2. RELATED WORK

The concept of a hole is well defined in Graph Theory [9]. From a graph-theoretic viewpoint [10], any chordless cycle of size at least 5 is a hole. A chord of a cycle C is defined as an edge connecting two non-consecutive vertices of C. A chordless cycle of a graph G is a cycle of length at least four in G that has no chord. Several algorithms exist to detect chordless cycles in simple graphs [11] with variations in definitions of holes and algorithmic complexity. Nikopoulous and Palios [10] describe an $O(|V| + |E|^2)$ serial algorithm which uses a specialized form of depth-first search (DFS) traversal attributed as (P_4-DFS) for detecting the existence of chordless cycles of length at least 5 in arbitrary graphs. Here, P_4 refers to a path of length 4, without a chord. The algorithm works analogous to the standard DFS

UNCORRECTED PROOFS

Security Comm. Networks 2010; **3**:1–10 © 2010 John Wiley & Sons, Ltd.
DOI: 10.1002/sec

[12], except that, in its general step instead of extending a $P_1 : \{a\}$ into $P_2 : \{a - b\}$ it tries to extend a $P_3 : \{a - b - c\}$ into $P_4 : \{a - b - c - d\}$. Our definition of hole is related to finding chordless paths; however, since our notion of hole is geometrical we need to include only those P_4's whose geometric embedding is non-intersecting. The exposition of our algorithm is geometrical since it makes use of the structure of a simple closed looped polygon to annotate the hole nodes.

More recently, Fang *et al*. [13] describe one of the first algorithms for hole detection in sensor networks. For each node p, they identify using a simple local conservative criteria if that node is *stuck*; a node is stuck if the circumcenter of at least one triangle including itself and a pair of adjacent neighbors lies outside the communication radii of p. Roughly, this means that p and the pair of nodes with respect to which it is stuck correspond to a direction which could be a potential local minimum for greedy forwarding. They define a hole as closed region bounded by a non self-intersecting polygonal loop and the one which on its boundary includes the stuck nodes. The polygonal loop is then defined to be the boundary of the hole. They use the geometric *right-hand rule* [14] starting at the stuck node p to discover a non-self-intersecting cycle that starts and ends at p. Our method does not worry about the cycle completion step—(though it can be easily incorporated as a verification step at an additional cost of a trip around the cycle); however, it still detects the holes as validated through the simulations. Our definition of a hole is slightly different from that of Reference [13]. We define a hole to be a simple polygonal curve of size at least 5. This definition subsumes the case of a simple, closed polygonal loops of size 5 or greater. However, our approach would exclude polygonal loops of size less than 4 contrary to the definition of Reference [13]. We believe, this is a reasonable assumption since a polygon of size 4 has diameter 2, and could arguably be not considered as a hole boundary. Funke [15] presents a hole detection algorithm assuming just topological information. The approach is based on selecting appropriate landmark nodes to seed the hole detection process, and then flooding the network infrequently for exploration. In contrast, our approach does require topological and geometrical information of 2-hop nodes but is restricted locally and therefore less amenable to network topological fluctuations.

In a related, but a completely different approach, Ghrist and Muhammad [16] describe the richness of simplicial complexes: the *Nerve complex* and the *Rips complex* to identify holes by means of homology—a branch from algebraic topology. Their solution is coordinate free; however, the criterion they develop to detect a Rips complex is centralized and computation takes into account arbitrary number of nodes. It is an open algorithmic problem to adapt these techniques from algebraic topology to compute the holes in a decentralized framework assuming no geographic information. We next develop our notion of a hole. The simulation experiments confirms that this coincides with the intuitive notion of voids in the sensor fields.

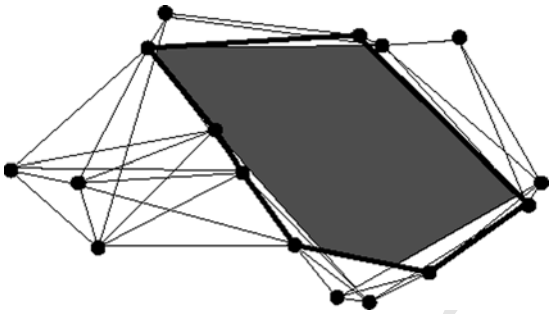


Figure 1. The solid boundary of a hole encapsulating a shaded face.

Definition (Arrangement of Segments)**.** *A subdivision of a plane into vertices, edges, and faces formed due to the intersection of line segments.*

Computing arrangements of line segments is an extensively researched area in computational geometry. Berg *et al*. [14] present a detailed exposition on the topic. To define a hole we make use of the notion of faces in the arrangements. The shaded face in Figure 1 shows a face in the arrangement.

Definition (Line Edge)**.** *Let $\bar{e}$ be an edge of a face $\bar{f}$ in the arrangement. Then, the line segment that contains $\bar{e}$ denoted as e is the line edge for $\bar{e}$.*

Remark. The edge of face is defined by its line edge and zero or more intersecting line edges in the arrangement.

Definition (Hole Edge)**.** *The line segment $\hat{e}$ of a minimal enclosed simple polygon $\hat{f}$ is a hole edge if it is mapped to the edge $\bar{e}$ of a face $\bar{f}$ in an arrangement.*

Definition (Hole)**.** *A hole is a minimal simple closed polygon of hole edges of size at least* 5 *that encloses in its interior a face in the arrangement of line segments such that no vertex of the face lies in the exterior of the polygon. The solid boundary in Figure* 1 *depicts a hole.*

We use the following notation in description that follows: the line segments or line edges as e, the corresponding edge of the faces of the arrangement as $\bar{e}$ and the hole edge of the minimal enclosing polygon that corresponds to $\bar{e}$ as $\hat{e}$. In what follows, the discovery of a hole will derive the mapping from a face edge $\bar{e}$ to a hole edge $\hat{e}$ using the line edge e and the line edges that intersect it to form $\bar{e}$. We refer to the first (tail) and second (head) vertices of an line edge in counter-clockwise orientation of the face as $e.first$ and $e.second$, respectively.

3. PROBLEM DEFINITION

3.1. High level description

The hole detection problem for sensor networks is simplified if we supplement the topological arrangement of

UNCORRECTED PROOFS

nodes with their geometrical information. Stated differently, detecting chordless cycles is easier if we know the relative clock wise orientation of all the immediate neighbors of a node. We believe that this is reasonable assumption considering the wide spread prevalence of GPS and accelerometer enabled mobile devices. We assume the sensor nodes can detect their self-coordinates and those of their neighbors with local mechanisms. A sufficient requirement is that a node is aware of the anti-clockwise ordering of all its neighbors. In practice, however, we only need to know the neighbors which are the part of a network overlay, since these nodes form the routing backbone of the sensor network and are typically of size O(1) per node. Moreover, the decision of the self-identification as a hole boundary node is based on the nodes no further than 2-hops from it. Let $G = (V, E)$ be a geometric graph induced by the distribution of nodes. Let the number of nodes be $n = |V|$. Any, two nodes v_1, v_2 are joined by an edge $e = (v_1, v_2) \iff d(v_1, v_2) \leq 1$, where we have normalized the uniform transmission/reception radii to a unit for the simplicity of exposition. We would interchangeably refer to the nodes as points and edges as lines. The alternate naming convention refers to the geometric embedding of the graphical entities. Each node v, maintains a counter-clockwise(CCW) ordering of its incident edges. The two consecutive edges in the CCW-ordering of incident of v together with the incident vertices form a *Cone* centered at v. The algorithm identifies *Cones* which satisfy a topological criteria and marks its nodes and edges as *hole* edges. If an edge is hole-marked, then so are its incident vertices. The set of hole edges and vertices induce the geometric hole-graph. Since, we require the holes to be simple polygons, the hole graph must be planar, i.e., no two hole edges intersect. In order to achieve that, at any given step of the protocol we consider only those edges which do not intersect an existing hole-marked edge.

3.2. Geometrical insights

We begin this section with remarks about some known geometric facts and lemmas which are key to the development of the hole detection algorithm.

Lemma 3.1 (Visibility)**.** *If two edges of a unit disk graph intersect, then there exists at least one vertex among the four end vertices which is visible to the rest.*

Definition (Visible Vertex of Intersection)**.** *The vertex among the four vertices of a pair of intersecting edges which is visible from the rest of the vertices is called a visible vertex of the intersection.*

Definition (Triangle of Intersection)**.** *Any visible vertex of an intersection forms a triangle between the endpoints of the edge which it is not incident on.*

Remark. There is at least one visible vertex per pair of intersecting edges (and at least one triangle of intersection) by Lemma 3.1. In case of a clique all four vertices in the intersection would be visible and there would be four triangles of the intersection.

Definition (Essential Edge)**.** *An intersecting edge e is essential only if it is not contained in any triangle.*

Remark. Essential edge can be determined locally.

Since an essential edge is intersecting and not contained in a triangle, removing the edge will possibly disconnect the graph induced by the vertices of the essential edge and the edges it intersects. Essential edges can occur only when the intersection is very skew. An intersecting edge e may be contained in a triangle of intersection with one edge and may not be contained in a triangle of intersection with some other edge—in that case it is not essential. e is essential only if it is not contained in any triangle. Refer to Figure 3 for the picture of the concepts developed in the section.

Lemma 3.2 (Essential Edge Intersection)**.** *Two essential edges do not intersect.*

Proof. If two edges $e_1 = (u_1, v_1), e_2 = (u_2, v_2)$ intersect, then by Lemma 3.1 at least one vertex, wlog say u_1 should be visible to the rest. In particular, u_1 is visible to u_2, v_2. Thus, u_1, u_2, v_2 form a triangle. Therefore $e_2 = (u_2, v_2)$ cannot be an essential edge by definition ■

Lemma 3.3 (Edge Intersecting Two Essential Edges)**.** *If there is a common intersecting edge between two essential edges, then the two essential edges are connected by an edge.*

Proof. Let $e = (u, v)$ intersect the essential edges $e_1 = (u_1, v_1), e_2 = (u_2, v_2)$. By Lemma 3.1 and 3.2, in the graph induced by the intersecting edges e is contained in a triangle and e_1 (similarly e_2) is not. wlog, let u_1 and u_2 be on the same side of line e. Then, $\angle uu_1v_1 > \frac{\pi}{3}, \angle vu_1v_1 > \frac{\pi}{3}$ and $\angle vu_2v_2 > \frac{\pi}{3}, \angle uu_2v_2 > \frac{\pi}{3}$. Thus, $\angle uu_1v > \frac{2\pi}{3}$ and $\angle uu_2v > \frac{2\pi}{3}$. Now $\angle uu_1u_2 = \angle uu_1v + \angle vu_1u_2 > \frac{2\pi}{3} > \frac{\pi}{2}$. Thus, $u - u_2$ is the longest side in the $\triangle uu_1u_2$. Since $d(u_1, u_2) < d(u, u_2)$ and (u, u_2) is an edge (a side of $\triangle uu_1v$) so is (u_1, u_2). ■

Remark. Lemma 3.2 and 3.3 imply that even after removing the edges intersecting the essential edges the graph induced on the vertices of essential edges and their intersecting edges is connected.

We present a constructive proof for the polygon that encloses the face of an arrangement in its interior. Consider a closed face $\bar{f}$ of an arrangement enclosed with the counter-clock wise sequence of edges $\bar{e_0}, \bar{e_1}, \ldots \bar{e_{n-1}}$ (indices modulo $n \geq 5$), with respective (possibly intersecting) line edges $e_0, e_1, \ldots, e_{n-1}$. If none of e_is were intersecting then the hole polygon would be just the sequence of e_is if ($n \geq 5$). However, consider a face edge $\bar{e_i}$ with its line edge e_i intersecting with e_{i-1} and e_{i+1} in CCW orientation. Based on the different scenarios and the order in which neighboring edges are mapped to hole edges $\bar{e_i}$

UNCORRECTED PROOFS

Security Comm. Networks 2010; **3**:1–10 © 2010 John Wiley & Sons, Ltd.
DOI: 10.1002/sec

would be mapped to a line edge close to the face boundary. The proof attempts to construct such mapping based on different scenarios of interaction between e_{i-1}, e_i, and e_{i+1}.

Theorem 3.4 (Construction of Hole)**.** *There exists a simple hole-polygon which tightly encloses the inward closed face of an arrangement.*

Proof. wlog, pick an ordering of line edges along a face of an arangement such that e_0 is a simple edge (non-intersecting with any e_i); else an essential edge or else if it intersects with e_{n-1}, let e_{n-1} be contained in the triangle of intersection. Note that if no such e_0 exists, then there are no essential edges and every consecutive edge belongs to a triangle of intersection and we can pick e_0 arbitrarily. In the course of the proof, we would construct the correspondence between the face edges $\bar{e_i}$ and the hole edge $\hat{e_i}$, which is selected from some line edge close to the face boundary. If the hole edge $\hat{e_i}$ is same as the line edge e_i we say that the correspondence is simple. The Figure 4 illustrates the different cases of the interaction between three consecutive line edges of a face of an arrangement, i.e., e_{i-1}, e_i, e_{i+1} the corresponding hole edges are marked with bi-directional arrows.

case ($i < 1$): Let

$$\hat{e_0} = e_0,$$

then,

$$\hat{e_1} = (e_0.second, e_1.second)$$

If e_1 follows e_0, then $\hat{e_1} = e_1$; otherwise e_1 intersects with e_0 and since e_1 is contained in the triangle of intersection (by choice of e_0), there exists an edge $(e_0.second, e_1.second)$ in the triangle.

case ($2 \leq i < n-1$): All the correspondences until index $2 \leq i < n-1$ have been defined. Consider edge $\bar{e_{i+1}}$

- if e_i is essential or simple then $\hat{e_i} = e_i$; since the prior mapped edges $\hat{e_j}$, $j < i$ would not intersect with e_i.
- If e_{i+1} is an essential edge, then by Lemma 3.2, e_i cannot be essential.

$$\hat{e_{i+1}} = e_{i+1}$$
$$\hat{e_i} = (\hat{e_{i-1}}.second, e_{i+1}.first)$$

There are two situations to consider in this case if e_{i-1} is essential, then, by 3.3 there is an edge connecting the two and in that case $\hat{e_i}.first = \hat{e_{i-1}}.second = e_{i-1}.second$. If e_{i-1} is not essential, then we can always keep $\hat{e_i}.first = e_i.first$ when we process e_i and set $\hat{e_i}.second = e_{i+1}.first$ as shown in Figure 4(a)

- Else e_{i+1} is not essential, again there are two situations to consider. If e_i is in the triangle of intersection, then

$$\hat{e_i} = (\hat{e_{i-1}}.second, e_i.second)$$

and

$$\hat{e_{i+1}} = (\hat{e_i}.second, e_{i+1}.second)$$

as shown in Figure 4(b) or

$$\hat{e_i} = \phi$$

and

$$\hat{e_{i+1}} = e_{i+1}$$

as depicted in Figure 4(c). Otherwise e_{i+1} is in the triangle of intersection and

$$\hat{e_i} = e_i$$
$$\hat{e_{i+1}} = (\hat{e_i}.second, e_{i+1}.second)$$

as shown in Figure 4(d)

case: ($i = n-1$). By choice of e_0, e_{n-1} cannot be essential by 3.2.

- If e_{n-2} is essential or a simple edge, then by 3.3 there is an edge which connects them

$$\hat{e_{n-1}} = (e_{n-2}.second, e_0.first)$$

- Else If e_{n-2} is in the triangle of intersection with e_{n-1}, then

$$\hat{e_{n-2}} = (\hat{e_{n-2}}.first, e_{n-1}.first)$$
$$\hat{e_{n-1}} = (e_{n-1}.first, e_0.first)$$

- Else e_{n-1} is in the triangle of intersection with e_{n-2}

$$\hat{e_{n-1}} = (\hat{e_{n-2}}.second, \hat{e_0}.first)$$

■

Corollary 3.5. *All line edges of a face $n \geq 5$ which are either simple or essential are hole edges.*

Corollary 3.6 (Chordless Enclosing Polygon)**.** *The polygon described by the oriented hole edges is closed and chordless inside the interior face.*

Proof. The choice of the hole edge is almost always along the line edge of a face in the arrangement or incident on one of the end vertices of adjacent line edges. Only in the case when two adjacent intersecting line edges are essential, the hole edge $\hat{e}$ is not incident on both vertices of the line edge e. Also, all vertices of the hole polygon are exterior or on the face boundary. If there exists a chord in the interior of the hole then that chord will be a line in the arrangement which will intersect the interior of the face. This contradicts the observation that the hole polygon encloses the face in its interior. The Theorem 3.4 ensures that the hole-polygon is closed. ■

UNCORRECTED PROOFS

Remark. The choice of the hole edge $\hat{e_i}$ depends exclusively on the pre-selection of atmost two adjacent hole edges $e_{i-1}^{\hat{}}$ and $e_{i+1}^{\hat{}}$.

3.3. Overview of the hole detection approach

The last subsection provided insights into existence of hole polygons tightly surrounding large faces of arrangements. In the subsequent section, we followup with a distributed, geometric hole-polygon detection algorithm. Consider consecutive edges w, x, y, z which form a chordless path of length 4. In that case, we can mark x, y as hole edges but not commit to w, z (because they can only be committed after looking at their other adjacent edges. This idea is used in designing the hole detection algorithm in Section 4 A node is marked as a hole by the first rule if there is a *Cup*-like structure centered on it. A *Cup* is essentially a chordless P_4, i.e., a path on 5 vertices with the hole-marked vertex at its center (see Figure 5 and the chordless P_4 path $u_1 - u_0 - v - w_0 - w_1$). Thus, identifying nodes like v is a pre-cursor to chordless-cycle detection, our approach uses local geometry to find such nodes. Unlike the approach by Reference [13] we do not search a cycle around marked nodes.

4. THE EMPTY I-CONE BASED HOLE DETECTION ALGORITHM

4.1. Defining graphical structures

There are two rules to mark a node as a hole. The first one, is more conservative and discovers the essential edges which are the must candidates if it is adjacent to a hole boundary. The second rule which is the local rule, attempts to mark the hole edges which do not conflict the first rule or the edges which are marked asynchronously by a node executing the same rule.

Definition (i-Cone)**.** *An i-Cone is a pair of successive edges in CCW ordering incident on a vertex v which do not intersect any previously hole-marked edge or an essential edge in the graph.*

Remark. We prefer not overloading the affine geometric concept of a cone, hence to differentiate from that structure we chose the prefixed term i-Cone. Note that the definition of i-Cone depends only on the local geometry of nodes adjacent to v.

Refer to Figure 5 for the rest of the section. Thus, the definition of i-Cones is hinged on the earlier choices made by the algorithm in marking hole edges. Let us denote as i-Cone (u_0, v, w_0), the i-Cone in which w_0 follows u_0 ($w_0 \neq u_0$) in the CCW orientation of edges about v. v is called the central vertex of the i-Cone (u_0, v, w_0). If the angle made at the central vertex in the CCW orientation (turning from u_0 to w_0 about v CCW) is less than π, then the i-Cone (u_0, v, w_0) is convex else it is concave. Cone (u_1, u_0, v) (Cone (v, w_0, w_1)) is defined as the *left* (resp. *right*) Cone of i-Cone (u_0, v, w_0) if it exists. For each convex i-Cone (u_0, v, w_0) we define an extended i-Cone $(u_0, v, w_0) =$ Cone $(u_1, u_0, v) \cup$ i-Cone $(u_0, v, w_0) \cup$ Cone (v, w_0, w_1). Note that u_1 is uniquely determined by v as its CCW neighbor around the central vertex u_0. Similarly, w_1 is uniquely determined as a CCW neighbor of v about the central vertex w_0. The hole detection algorithm is based on finding specific extended i-Cones, which we refer to as an *empty* i-Cones. An i-Cup (u_0, v, w_0) is the one in which the graph induced by the vertex set $\{u_1, u_0, v, w_0, w_1\}$. An i-Cone is empty if the i-Cup is a simple chordless path or a chordless cycle. In other words, the only permissible induced graph between the vertex set of an empty i-Cup (u_0, v, w_0) is the chordless path $u_1 - u_0 - v - w_0 - w_1$ or the chordless cycle $u_1 - u_0 - v - w_0 - w_1 - u_1$. v is called the central vertex of the i-Cup (u_0, v, w_0).

(1) **Essential Edge Rule** mark all the essential edges which are determined locally. The essential edges are those which are intersecting and yet not contained in a triangle. The next rule would never choose edges which intersect with the essential edges
(2) **Empty i-Cone Rule**. i-Cone (u, v, w) is *empty* if it is concave or has an i-Cup (u, v, w) which is chordless. The i-Cone (u, v, w) edges must not intersect a previously marked hole edge or an essential edge.

Algorithm 4.1 Hole detection

```
Input G(V, E, VLF : V → (x, y))
    BEGIN
 1: for all v ∈ V do
 2:   if v.hasConesGreaterThan(π/3) then
 3:     if v.acquireTwoHopNeighborLocks() then
 4:       for all C ∈ v.getEmptyIConesGreaterThan(π/3) do
 5:         C.markHole()
 6:       end for
 7:       v.releaseTwoHopNeighborLocks()
 8:     end if
 9:   end if
10: end for
    END
```

4.2. Algorithm description

The lines 2–4 in Algorithm 4.1 correspond to marking holes by the empty i-Cone rule. Observe that, based on unit disk graph geometry there can be at most 5 convex Cones around any central vertex each of which could potentially form

UNCORRECTED PROOFS

Security Comm. Networks 2010; **3**:1–10 © 2010 John Wiley & Sons, Ltd.
DOI: 10.1002/sec

Algorithm 4.2 V::getEmptyIConesGreaterThan(double θ)

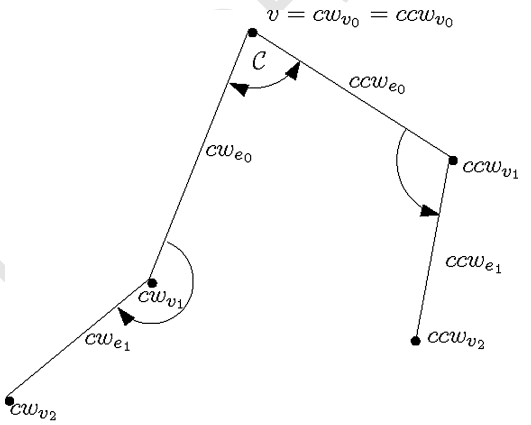
Input $\mathcal{G}(V, E, \mathcal{VLF}: V \rightarrow (x, y))$

BEGIN

1: v.ccwSortIncidentEdges()
2: $\mathcal{C} \leftarrow v$.getIncidentIConesGreaterThan(θ)
3: **for all** $c \in \mathcal{C}$ **do**
4: **if** c.isConcave() **then**
5: **continue**// *c is an empty i-Cone so do not remove*
6: **Else** //c is convex
7: $cw_{e_0} \leftarrow c$.getRightEdge()
8: $ccw_{e_0} \leftarrow c$.getLeftEdge()
9: $u_0 \leftarrow v$.getOpposite(cw_{e_0})
10: $w_0 \leftarrow v$.getOpposite(ccw_{e_0})
11: $cw_{e_1} \leftarrow u_0$.getNextCWEdgeTo(cw_{e_0})
12: $ccw_{e_1} \leftarrow w_0$.getNextCCWEdgeTo(ccw_{e_0})
13: $u_1 \leftarrow u_0$.getOpposite(cw_{e_1})
14: $w_1 \leftarrow w_0$.getOpposite(ccw_{e_1})
15: **if** $(u_1, u_0, v, w_0, w_1)$is NOT chordless **then**
16: $\mathcal{C} \leftarrow \mathcal{C} \setminus c$ //remove c from the set $\mathcal{C}$
17: **end if**
18: **end if**
19: **end for**
20: **return** $\mathcal{C}$

END

an i-Cup (u, v, w) or at most one concave i-Cone (u, v, w). We refer to Algorithm 4.2 for the further discussion on the computation of empty i-Cones. Line 1–2 compute the candidates for empty i-Cones. These are all i-Cones which are greater than θ. The rest of the Algorithm 4.2 simply prunes the i-Cones which do not satisfy the empty i-Cup definition. Lines 4–5 deal with concave i-Cones. Note that by definition any concave i-Cone (u, v, w) is an empty i-Cone regardless of whether a u–w edge exists or not as shown in the Figure 2. The intuition being that sweeping



$$cw_{nodes} = \{cw_{v_1}\} \cup \{cw_{v_2}\}$$
$$ccw_{nodes} = \{ccw_{v_1}\} \cup \{ccw_{v_2}\}$$

Figure 2. Hole marking the vertex v using the first criteria.

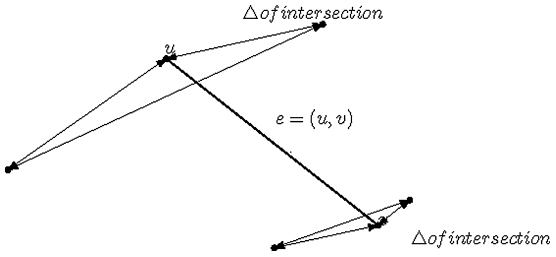


Figure 3. Visible vertices u, v and the essential edge e.

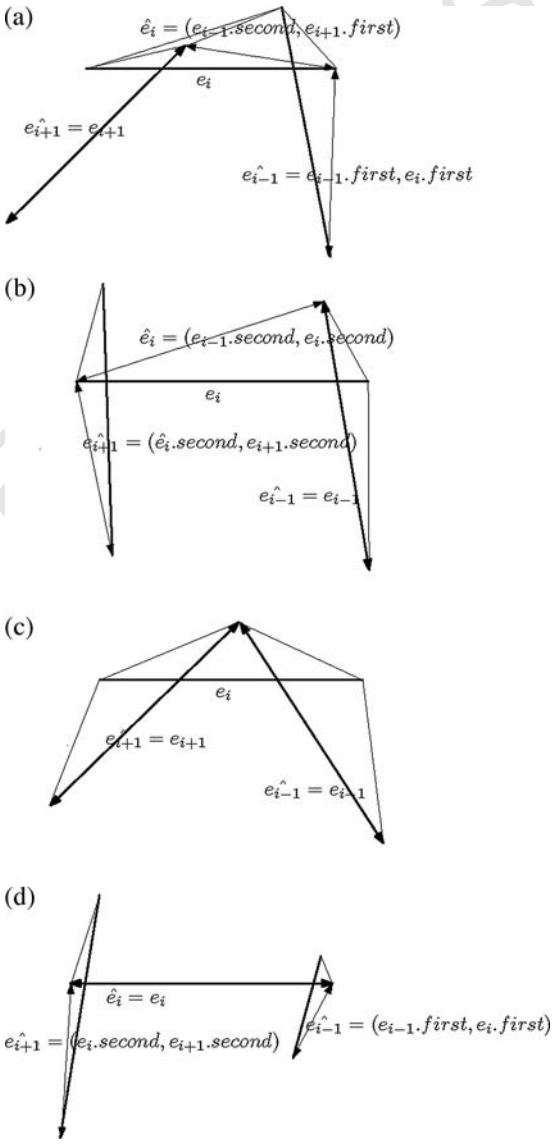


Figure 4. Hole construction proof (hole edges are indicated by bidirectional arrows, line edges are shown in bold).

u to w about v in the CCW sense does not encounter any edge since the angle is concave. Lines 7–16, prune convex i-Cones which do not correspond to chordless i-Cups. Let $\mathcal{C}$ be a i-Cone with central angle greater than $\frac{\pi}{3}$ centered at v

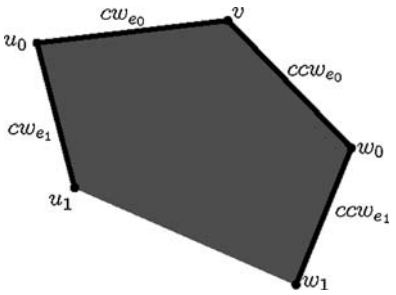


Figure 5. $u_1 - u_0 - v - w_0 - w_1$ is a chordless cycle.

(see Figure 2). Let $cw_{e_0}(ccw_{e_0})$, be the pair of ordered Cone edges in the clock-wise (resp. counter-clockwise sense). In the for loop on lines 9–14, the algorithm computes the edges $cw_{e_1}(ccw_{e_1})$, as the CW (resp. CCW) neighbor of the edges $cw_{e_0}(ccw_{e_0})$, about the vertex u_0 $(resp. w_0)$. The CW (resp. CCW) edges which intersect a previously marked hole edge or an essential edge are filtered (and the next one in order are considered). This traversal is similar in fashion to the the edge traversal along the boundary of a simple planar polygon using the familiar *right hand rule*. Thus, if the Empty i-Cup exists or equivalently the graph induced by the nodes $u_1 - u_0 - v - w_0 - w_1$ is a chordless path or a chordless cycle, then the i-Cone is empty otherwise it is removed from the set of i-Cones on line 16. If such an empty i-Cone (u, v, w) exists, then the Algorithm 4.1 (lines 4–6) marks u, v, w as hole vertices and (u, v) and (v, w) as hole edges. Since the definition of empty i-Cups is relative to convex i-Cones, it can be easily computed once each node keeps an updated list of empty i-Cones centered on it.

4.3. Algorithm psuedocode

We present the psuedocode for hole detection in Algorithm 4.1. The sub-routine to compute empty i-cones is presented in Algorithm 4.2

5. EXPERIMENTAL RESULTS

We tested the *Empty i-Cone* based hole marking algorithm by simulating random geometric graphs on varying distributions of nodes. We used the JUNG (Java Universal Network Graph) [17] framework for extending the graph library for geometric graphs. We verified the mechanisms for hole information updation by enforcing mobilty on a subset of nodes. The results prove the self-configuring behavior of the hole-detection protocol on different node densities. The visualization snapshots for the simulation are presented in Figures 6 and 7. We simulated the uniform hole distributions for 300 and 600 nodes in a 10×10 square unit area. We also created the non-uniform distributions for 300 and 600 nodes by inserting voids in the same region. The hole detection algorithm identifies the external boundary of nodes correctly. The visualization also demonstrates that the voids

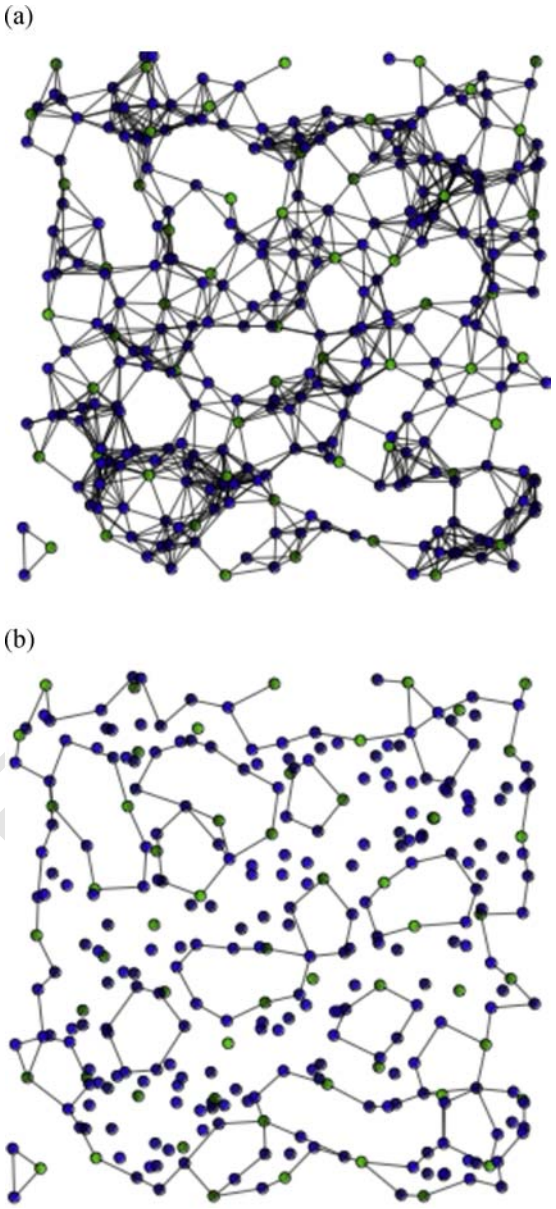


Figure 6. Simulation results for 300 nodes in 10×10 square units: (a) Original distribution and (b) Hole boundary distribution.

in the original distribution are correctly identified with a closed polygonal loop. The hole boundary identified with closed polygonal edges in Figures 6(b) and 7(b), respectively capture the geometry of the void space reasonably well.

6. SECURITY IMPLICATIONS WITH HOLE DETECTION

The hole detection algorithm requires geometrical coordinates of its 2-hop neighbors. Since, the empty i-Cone algorithm identifies candidate convex cones at each node,

UNCORRECTED

Security Comm. Networks 2010; **3**:1–10 © 2010 John Wiley & Sons, Ltd.
DOI: 10.1002/sec

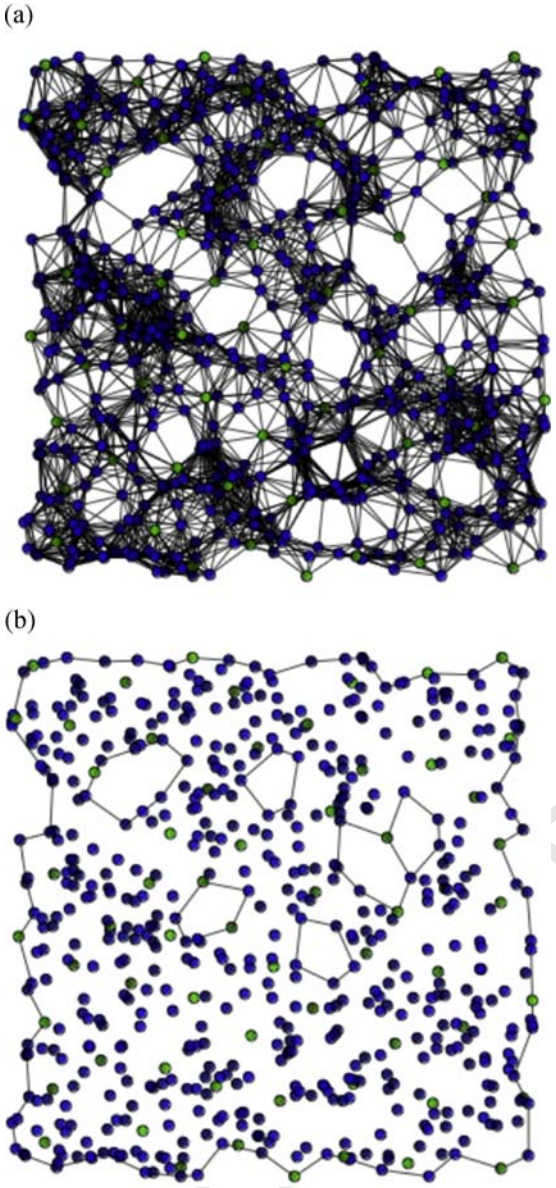


Figure 7. Simulation results for 600 nodes in 10 × 10 square units: (a) Original distribution and (b) Hole boundary distribution.

it is sufficient to authenticate the convex cones identified by its neighbors, since the empty *i*-Cone at *v* is the union of *i*-Cones of its CW and CCW neighbors. The hole detection algorithm relies heavily on correctly identifying essential edges and circular orientation of edges incident on a node and the link authentication can ascertain the geometric coordinates accurately. In this sense, link layer security mechanisms are sufficient to detect the local *i*-Cones securely. Furthermore, if a link is deemed insecure by a node, then it can eliminate the node and the edge from its local graph during hole computation. Marking a hole edge requires synchronization between its incident vertices, only secure links can be marked as hole edges. However, since the hole detection framework does not verify the closure of polygons, insecure links could create disconnected chordless paths in the network, especially so if there are insecure nodes along the boundary of the face. This is a serious deficiency of the local algorithm that eliminates insecure edges. If we assume that the insecure nodes advertise their coordinates precisely and consistently then we can instead include them in computing the hole boundary. So now we have a hole boundary of secure and insecure nodes and links. We would then need a non-local multi-party security algorithm to maintain the integrity of the hole. Depending on the size of the insecure nodes in the hole this may or may not be practical. The secret has to be shared by atleast a significant fraction of nodes on the hole boundary and the nodes which share that secret should verify the integrity of the choices made by nodes on the hole boundary.

Sybil attacks [18], where an adversary poses as an imposter by advertising different or falsified location coordinates to different nodes can potentially attack geography based protocols. In this situation, the secure protocol will need a third party verification of node identities and certificates verifying their geographic positions. This approach will however require a more detailed analysis on the security aspects of the algorithm and a modified protocol can be a subject matter of discussion outside the scope of this paper. We believe a pure topological approach [10] on detection of hole boundary will alleviate the security concerns stirred from falsfied geometry. However, the known approaches are non-local.

7. CONCLUSION

We have presented a local, distributed hole detection algorithm. We implemented the algorithm and verified it using random geometric graphs for both uniform and non-uniform distributions. We formalized the notion of a hole deriving ideas from computational geometry. The chapter unifies geometry and graph-theoretic ideas to find a closed polygonal boundary encapsulating voids in an ad hoc network. Our method for hole detection is simpler, localized, and performs well on experimental simulations. We discussed the relevance of the hole detection framework for topology-awareness, routing, and the potential security pitfalls.

REFERENCES

1. Reason JM, Rabaey JM. A study of energy consumption and reliability in a multi-hop sensor network. *SIGMOBILE Mobile Computing and Communications Review* 2004; **8**(1): 84–97. DOI:http://doi.acm.org/10.1145/980159.980170
2. Yener B, Magdon-Ismail M, Sivrikaya F. Joint problem of power optimal connectivity and coverage in wireless sensor networks. *Wireless Networks* 2007; **13**(4): 537–550. DOI:http://dx.doi.org/10.1007/s11276-006-5875-0

UNCORRECTED PROOFS

3. Bose P, Morin P, Stojmenovic I, Urrutia J. Routing with guaranteed delivery in ad hoc wireless networks. *Wireless Networks* 2001; **7**(6): 609–616.
4. Karp B, Kung HT. GPSR: Greedy perimeter stateless routing for wireless networks. *ACM/IEEE Symposium International Conference on Mobile Computing and Networking (Mobicom)*, 2000; 243–254.
5. Kranakis E, Singh H, Urrutia J. Compass routing on geometric networks. *Proceedings of the 11th Canadian Conference on Computational Geometry*, Vancouver, 1999; 51–54.
6. Albano M, Chessa S, Nidito F, Pelagatti S. Geographic hash tables with QoS in non uniform sensor networks. *ACM MobiHoc 2006 Poster Proceedings*, Firenze, Italy, 2006.
7. Das S, Pucha H, Hu YC. Performance comparison of scalable location services for geographic ad hoc routing. *Proceedings of IEEE INFOCOM*, 2005; 120–130.
8. Li J, Jannotti J, Couto DD, Karger D, Morris R. A scalable location service for geographic ad hoc routing. *Sixth Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)*, August 2000; 120–130.
9. Diestel R. *Graph Theory* (3rd edn). Springer-Verlag Heidelberg: Newyork, 2005.
10. Nikolopoulos S, Palios L. Hole and antihole detection in graphs. *Proceedings of the 15th Annual ACM-SIAM Symposium on Discrete Algorithms*, New Orleans, Louisiana, 2004; 850–859.
11. Spinrad J. Finding large holes. *Information Processing Letters* 1991; **39**(4): 227–229.
12. Cormen TH, Leiserson CE, Rivest RL, Stein C. *Introduction to Algorithms* (2nd edn). MIT Press: Cambridge, MA, 2001.
13. Fang Q, Gao J, Guibas L. Locating and bypassing routing holes in sensor networks. *The 23rd Annual Joint Conference of the IEEE Computers and Communications Society (Infocom)*, Vol. 4, Hong Kong, PRC, 2004; 2458–2468.
14. de Berg M, van Kreveld M, Overmars M, Schwarzkopf O. *Computational Geometry: Algorithms and Applications*. Springer-Verlag New York, Inc.: Secaucus, NJ, 1997.
15. Funke S. Topological hole detection in wireless sensor networks and its applications. *DIALM-POMC '05: Proceedings of the 2005 Joint Workshop on Foundations of Mobile Computing*, ACM: New York, 2005; 44–53. DOI:http://doi.acm.org/10.1145/1080810.1080819
16. Ghrist R, Muhammad A. Coverage and hole-detection in sensor networks via homology. *Fourth International Symposium on Information Processing in Sensor Networks (ISPN)*, Los Angeles, California, 2005; 254–260.
17. The JUNG Project Website. 2006. Available at: http://jung.sourceforge.net/
18. Karlof C, Wagner D. Secure routing in wireless sensor networks: attacks and countermeasures. In *First IEEE International Workshop on Sensor Network Protocols and Applications*, 2002; 113–127.

UNCORRECTED PROOFS

A. Shirsat* and B. Bhargava ..... xxx–xxx

Local geometric algorithm for hole boundary detection in sensor networks

Q1

UNCORRECTED PROOFS

Author Query Form (SEC/271)

Special Instruction: Author please include responses to queries with your other corrections and return by e-mail.

Q1: Author: Please provide a suitable figure (abstract diagram or illustration selected from the manuscript or an additional 'eye-catching' figure) and a short 'GTOC' abstract (maximum 80 words or 3 sentences) summarizing the key findings presented in the paper for Table of Content (TOC) entry.

Q2: Author: Please verify the presentation of affiliations and correspondence address.

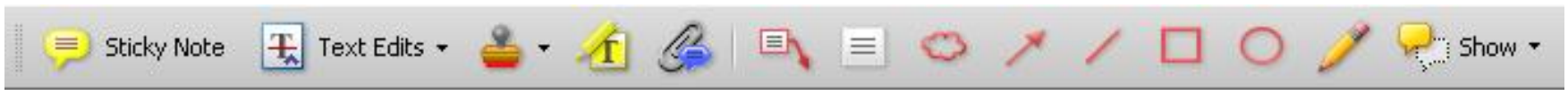
Q3: Author: Sections 3 and 6 are not described in this paragraph. Please describe them in sequence.

Q4: Author: Please check the usage of 'in the for loop'.

UNCORRECTED PROOFS

Required Software to eAnnotate PDFs: Adobe Acrobat Professional or Acrobat Reader (version 8.0 or above).
The Latest version of Acrobat Reader is free: http://www.adobe.com/products/acrobat/readstep2.html

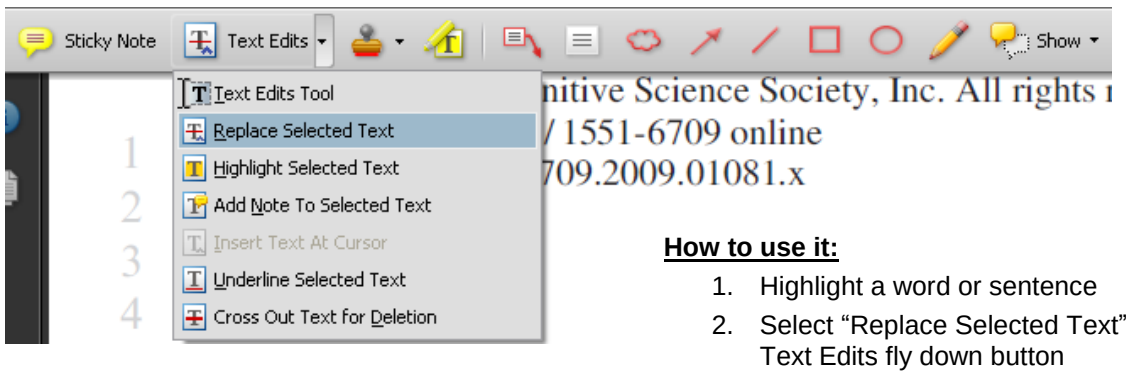
Once you have Acrobat Reader 8, or higher, open on your PC you should see the Commenting Toolbar:

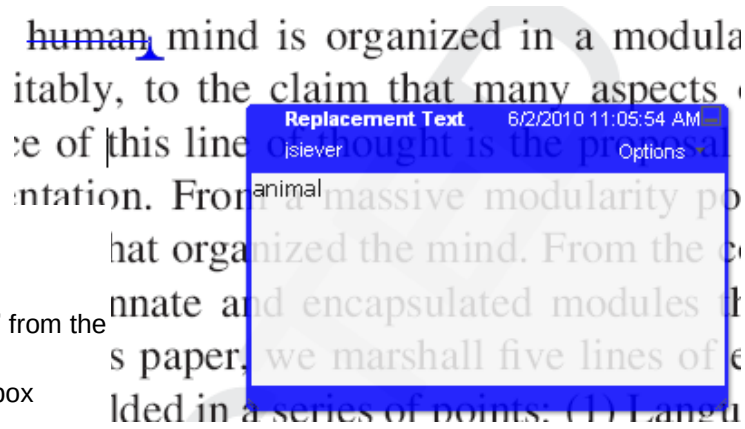


****(If the above toolbar does not appear automatically go to *Tools>Comment & Markup>Show Comment & Markup Toolbar*)****

1. Replacement Text Tool — For replacing text.

Strikes a line through text and opens up a replacement text box.



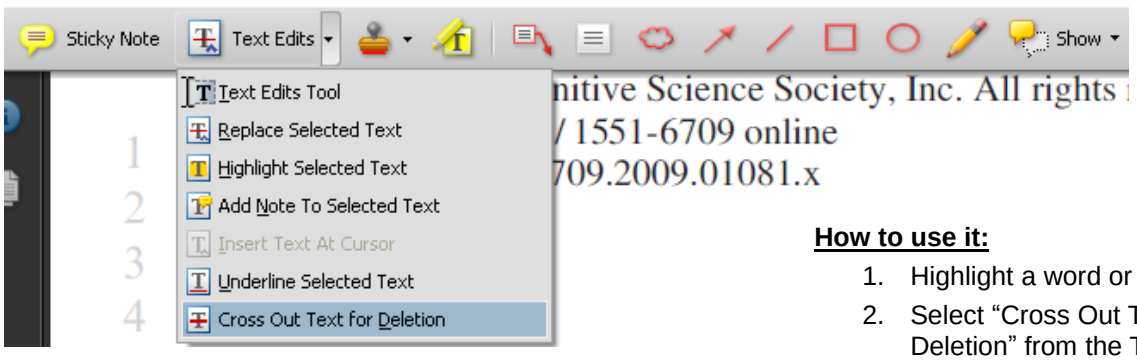


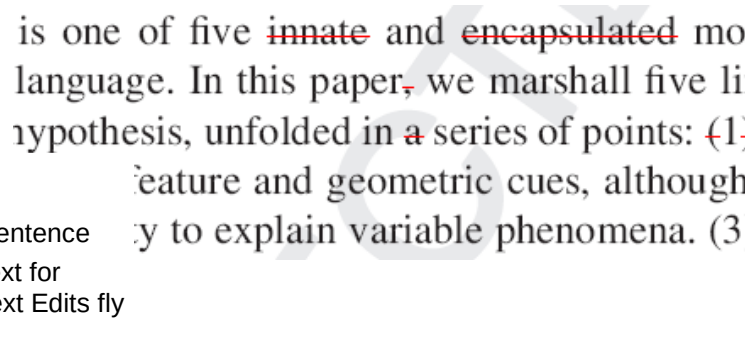
How to use it:

1. Highlight a word or sentence
2. Select “Replace Selected Text” from the Text Edits fly down button
3. Type replacement text in blue box

2. Cross-out Text Tool — For deleting text.

Strikes a red line through selected text.



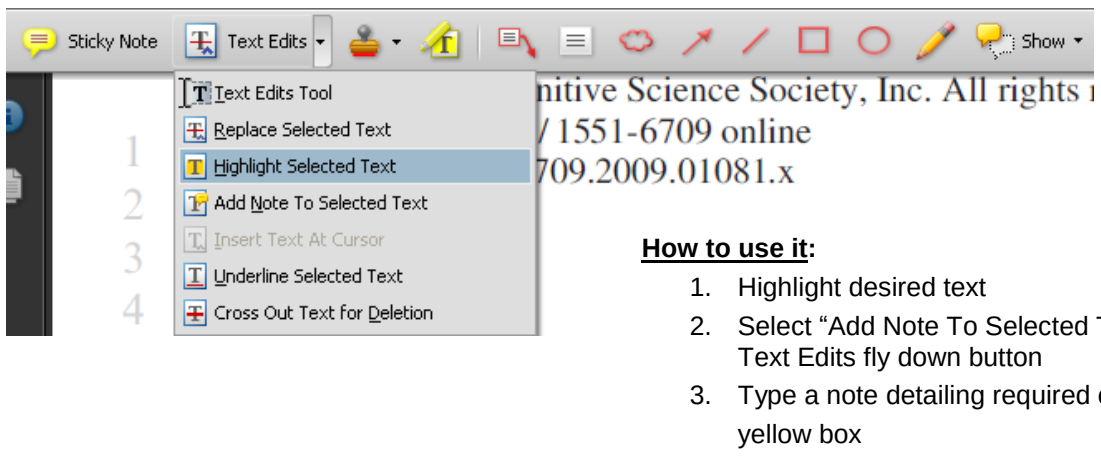


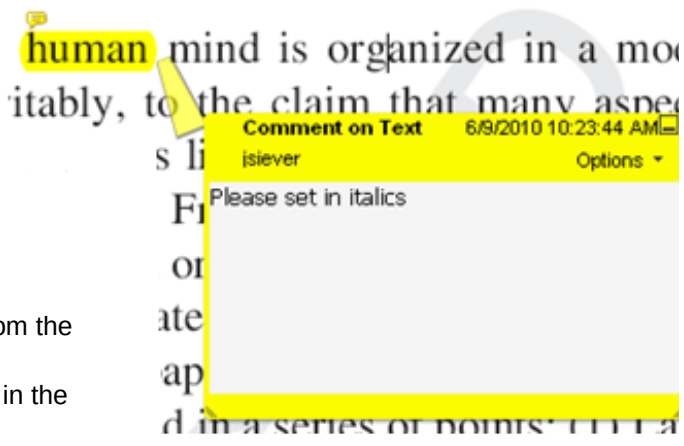
How to use it:

1. Highlight a word or sentence
2. Select “Cross Out Text for Deletion” from the Text Edits fly down button

3. Highlight Tool — For highlighting a selection to be changed to bold or italic.

Highlights text in yellow and opens up a text box.





How to use it:

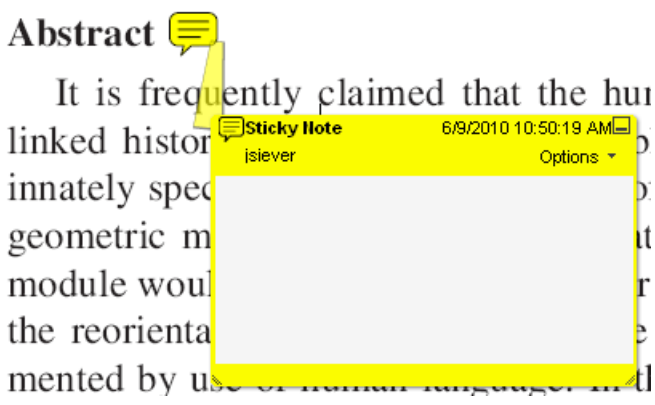
1. Highlight desired text
2. Select “Add Note To Selected Text” from the Text Edits fly down button
3. Type a note detailing required change in the yellow box

4. Note Tool — For making notes at specific points in the text

Marks a point on the paper where a note or question needs to be addressed.

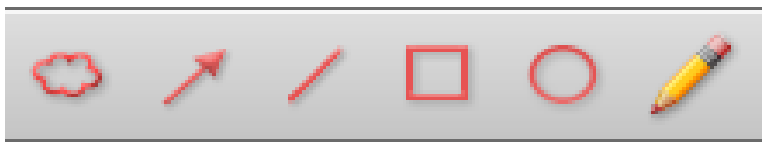
How to use it:

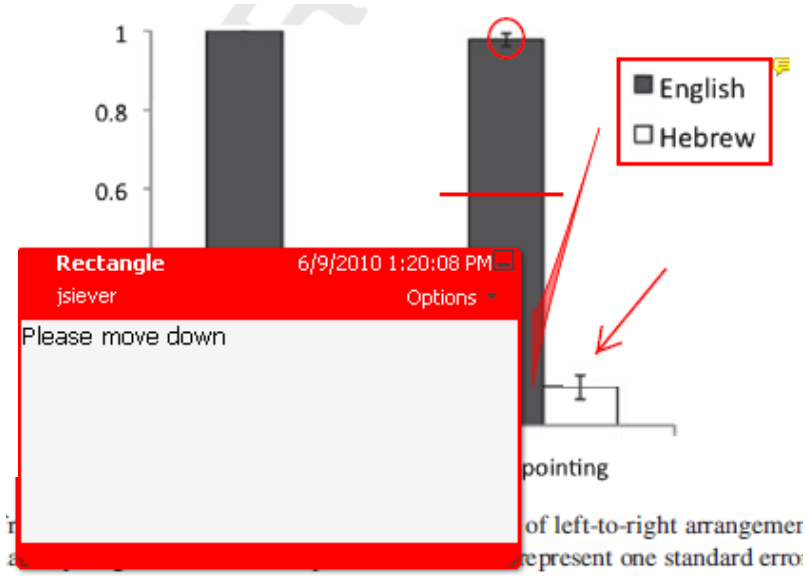
1. Select the Sticky Note icon from the commenting toolbar
2. Click where the yellow speech bubble symbol needs to appear and a yellow text box will appear
3. Type comment into the yellow text box



5. Drawing Markup Tools — For circling parts of figures or spaces that require changes

These tools allow you to draw circles, lines and comment on these marks.



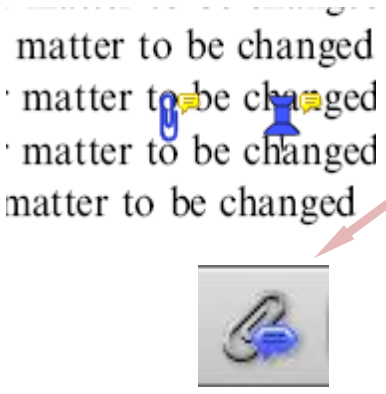


How to use it:

1. Click on one of shape icons in the Commenting Toolbar
2. Draw the selected shape with the cursor
3. Once finished, move the cursor over the shape until an arrowhead appears and double click
4. Type the details of the required change in the red box

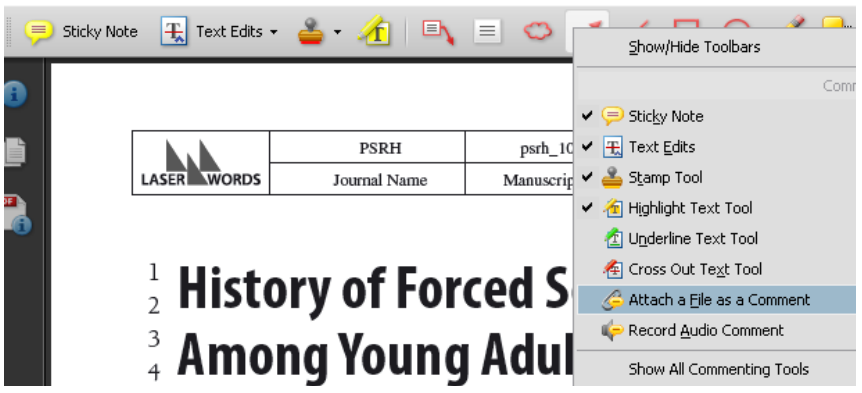
6. Attach File Tool — For inserting large amounts of text or replacement figures as a files.

Inserts symbol and speech bubble where a file has been inserted.

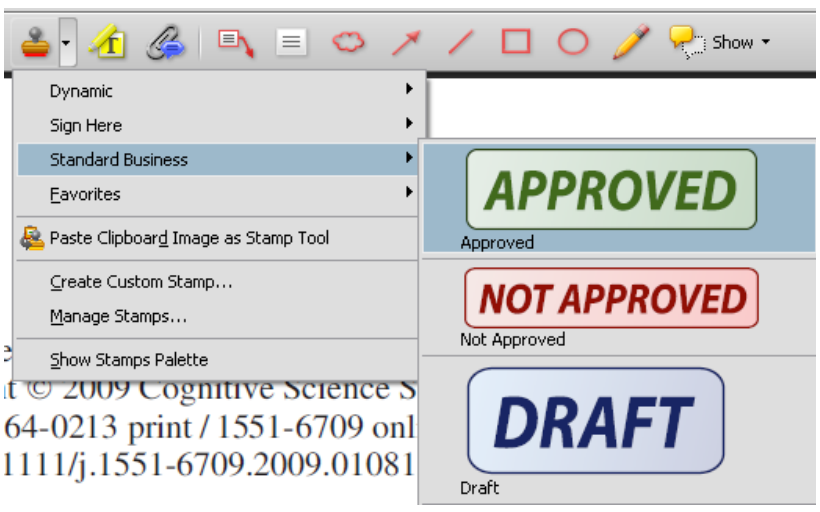


How to use it:

1. Right click on the Commenting Toolbar
2. Select “Attach a File as a Comment”
3. Click on paperclip icon that appears in the Commenting Toolbar
4. Click where you want to insert the attachment
5. Select the saved file from your PC or network
6. Select type of icon to appear (paperclip, graph, attachment or tag) and close

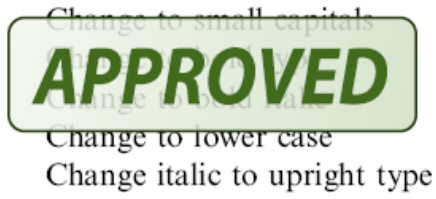


7. Approved Tool (Stamp) — For approving a proof if no corrections are required.



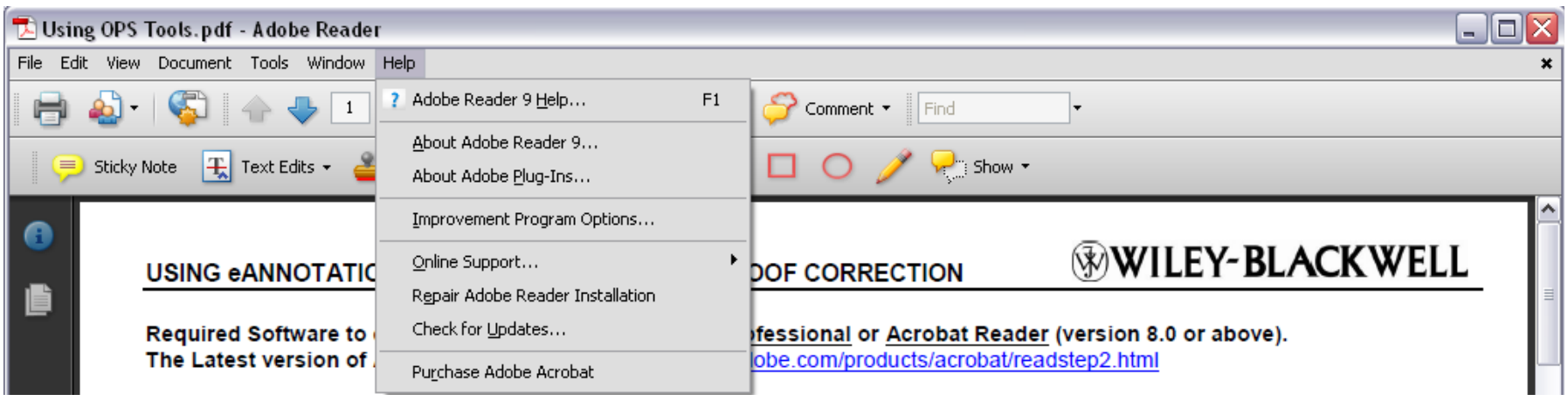
How to use it:

1. Click on the Stamp Tool in the toolbar
2. Select the Approved rubber stamp from the ‘standard business’ selection
3. Click on the text where you want to rubber stamp to appear (usually first page)



Help

For further information on how to annotate proofs click on the Help button to activate a list of instructions:



WILEY AUTHOR DISCOUNT CLUB

We would like to show our appreciation to you, a highly valued contributor to Wiley's publications, by offering a **unique 25% discount** off the published price of any of our books*.

All you need to do is apply for the **Wiley Author Discount Card** by completing the attached form and returning it to us at the following address:

The Database Group (Author Club)
John Wiley & Sons Ltd
The Atrium
Southern Gate
Chichester
PO19 8SQ
UK

Alternatively, you can **register online** at www.wileyeurope.com/go/authordiscount
Please pass on details of this offer to any co-authors or fellow contributors.

After registering you will receive your Wiley Author Discount Card with a special promotion code, which you will need to quote whenever you order books direct from us.

The quickest way to order your books from us is via our European website at:

http://www.wileyeurope.com

Key benefits to using the site and ordering online include:

- Real-time SECURE on-line ordering
- Easy catalogue browsing
- Dedicated Author resource centre
- Opportunity to sign up for subject-orientated e-mail alerts

Alternatively, you can order direct through Customer Services at:
cs-books@wiley.co.uk, or call +44 (0)1243 843294, fax +44 (0)1243 843303

So take advantage of this great offer and return your completed form today.

Yours sincerely,

Verity Leaver
Group Marketing Manager
author@wiley.co.uk

*TERMS AND CONDITIONS

This offer is exclusive to Wiley Authors, Editors, Contributors and Editorial Board Members in acquiring books for their personal use. There must be no resale through any channel. The offer is subject to stock availability and cannot be applied retrospectively. This entitlement cannot be used in conjunction with any other special offer. Wiley reserves the right to amend the terms of the offer at any time.

REGISTRATION FORM
For Wiley Author Club Discount Card

To enjoy your 25% discount, tell us your areas of interest and you will receive relevant catalogues or leaflets from which to select your books. Please indicate your specific subject areas below.

Accounting [ ]
- Public [ ]
- Corporate [ ]

Chemistry [ ]
- Analytical [ ]
- Industrial/Safety [ ]
- Organic [ ]
- Inorganic [ ]
- Polymer [ ]
- Spectroscopy [ ]

Encyclopedia/Reference [ ]
- Business/Finance [ ]
- Life Sciences [ ]
- Medical Sciences [ ]
- Physical Sciences [ ]
- Technology [ ]

Earth & Environmental Science [ ]

Hospitality [ ]

Genetics [ ]
- Bioinformatics/ Computational Biology [ ]
- Proteomics [ ]
- Genomics [ ]
- Gene Mapping [ ]
- Clinical Genetics [ ]

Medical Science [ ]
- Cardiovascular [ ]
- Diabetes [ ]
- Endocrinology [ ]
- Imaging [ ]
- Obstetrics/Gynaecology [ ]
- Oncology [ ]
- Pharmacology [ ]
- Psychiatry [ ]

Non-Profit [ ]

Architecture [ ]

Business/Management [ ]

Computer Science [ ]
- Database/Data Warehouse [ ]
- Internet Business [ ]
- Networking [ ]
- Programming/Software Development [ ]
- Object Technology [ ]

Engineering [ ]
- Civil [ ]
- Communications Technology [ ]
- Electronic [ ]
- Environmental [ ]
- Industrial [ ]
- Mechanical [ ]

Finance/Investing [ ]
- Economics [ ]
- Institutional [ ]
- Personal Finance [ ]

Life Science [ ]

Landscape Architecture [ ]

Mathematics Statistics [ ]

Manufacturing [ ]

Materials Science [ ]

Psychology [ ]
- Clinical [ ]
- Forensic [ ]
- Social & Personality [ ]
- Health & Sport [ ]
- Cognitive [ ]
- Organizational [ ]
- Developmental & Special Ed [ ]
- Child Welfare [ ]
- Self-Help [ ]

Physics/Physical Science [ ]

Please complete the next page /

I confirm that I am (*delete where not applicable):

a **Wiley** Book Author/Editor/Contributor* of the following book(s):
ISBN:
ISBN:

a **Wiley** Journal Editor/Contributor/Editorial Board Member* of the following journal(s):

SIGNATURE: …… Date: ……

PLEASE COMPLETE THE FOLLOWING DETAILS IN BLOCK CAPITALS:

TITLE: (e.g. Mr, Mrs, Dr) …… FULL NAME: ……

JOB TITLE (or Occupation): ……

DEPARTMENT: ……

COMPANY/INSTITUTION: ……

ADDRESS: ……

……

TOWN/CITY: ……

COUNTY/STATE: ……

COUNTRY: ……

POSTCODE/ZIP CODE: ……

DAYTIME TEL: ……

FAX: ……

E-MAIL: ……

YOUR PERSONAL DATA
We, John Wiley & Sons Ltd, will use the information you have provided to fulfil your request. In addition, we would like to:

1. Use your information to keep you informed by post of titles and offers of interest to you and available from us or other Wiley Group companies worldwide, and may supply your details to members of the Wiley Group for this purpose.
   [ ] Please tick the box if you do **NOT** wish to receive this information

2. Share your information with other carefully selected companies so that they may contact you by post with details of titles and offers that may be of interest to you.
   [ ] Please tick the box if you do **NOT** wish to receive this information.

E-MAIL ALERTING SERVICE
We also offer an alerting service to our author base via e-mail, with regular special offers and competitions. If you **DO** wish to receive these, please opt in by ticking the box [ ].

If, at any time, you wish to stop receiving information, please contact the Database Group (databasegroup@wiley.co.uk) at John Wiley & Sons Ltd, The Atrium, Southern Gate, Chichester, PO19 8SQ, UK.

TERMS & CONDITIONS
This offer is exclusive to Wiley Authors, Editors, Contributors and Editorial Board Members in acquiring books for their personal use. There should be no resale through any channel. The offer is subject to stock availability and may not be applied retrospectively. This entitlement cannot be used in conjunction with any other special offer. Wiley reserves the right to vary the terms of the offer at any time.

PLEASE RETURN THIS FORM TO:
Database Group (Author Club), John Wiley & Sons Ltd, The Atrium, Southern Gate, Chichester, PO19 8SQ, UK author@wiley.co.uk
Fax: +44 (0)1243 770154