This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.

Efficient, decentralized computation of the topology of spatial regions

Matt Duckham, Doron Nussbaum, Jörg-Rüdiger Sack, Nicola Santoro

***Abstract*—The capability to query the topology of spatial regions is fundamental to today's centralized spatial computing systems, like spatial databases and GIS. By contrast, this paper explores *decentralized* algorithms for computing the topology of spatial regions in wireless sensor networks. The approach generates global topological information about regions, using only the local knowledge of nodes and their immediate network neighbors aggregated up through spatial boundary structures. Using three basic boundary structures (boundary nodes, boundary cycles, and boundary orientation), a family of decentralized algorithms is defined that can respond efficiently to snapshot queries about the topology of spatial regions, including containment and adjacency queries. The communication complexity of the algorithm is $O(n)$ for realistic inputs. Empirical investigation of the performance of the approach, using simulation, also confirms the efficiency, scalability, and robustness of this approach.**

***Index Terms*—qualitative spatial reasoning, containment, adjacency, decentralized spatial computing, geosensor networks, wireless sensor networks**

I. Introduction

Computation in distributed systems increasingly occurs *somewhere*, with that location being integral to the computational process itself. For example, geographical wireless sensor networks (sometimes called *geosensor networks* [1]) are increasingly being tasked to respond to queries not only about point locations, but patterns and events with spatial and temporal extents. Many of the existing approaches to complex spatial queries in geosensor networks adopt a centralized approach to computation, where global spatial data is collated by the network but processed by a conventional information system, like a spatial database or GIS.

By contrast, this paper is concerned with problems of *decentralized* spatial computing (DeSC), where local spatial data is processed in the network itself, with no centralized or global control. Decentralization can help to improve the scalability of geosensor networks, for example increasing network longevity—a vital consideration in applications like long-term environmental monitoring (e.g., [2]). Specifically, this paper explores the design of efficient decentralized algorithms for responding to "snapshot" queries about the topological relationships between spatial regions, such as whether two regions are connected, adjacent, or contain one another. The regions are assumed to be derived from sensing of underlying environmental phenomena, like "hot spots," "wet regions," "contours," or "pollution clouds."

In any decentralized computing system, no one system component has access to the entire system state [3]. In a DeSC system, the location of nodes places additional spatial constraints on the generation and movement of information. For example, nodes in a geosensor network typically sense information only about their immediate spatial environment, and the energy required to communicate between nodes increases with spatial/network distance.

Queries about objects with spatial extents (like regions) are especially challenging because of these spatial constraints; satisfying complex spatial queries typically requires the combination of information captured and stored at distal locations. On the other hand, spatial information is inherently *autocorrelated* ("nearby things are more related," Tobler's so-called first law of geography). Thus, opportunities for efficient decentralized algorithms may exist because proximal nodes are more likely to be more relevant in answering spatial queries.

The central innovation in this paper is the definition of decentralized region *boundary* structures, which form the basis for efficient in-network spatial data aggregation. Section III formally defines the three levels of boundary structure required for topological queries: boundary nodes, boundary cycles, and boundary orientation. Section IV presents decentralized algorithms for constructing these boundary structures. Section V shows how these basic algorithms can be combined to respond to common topological queries, in particular containment and adjacency of regions. Section VI presents an empirical analysis of the performance of the algorithms using simulation, focusing on the efficiency, scalability, and robustness of the algorithms, before section VII concludes the paper.

II. Background

Research into the application of decentralized algorithms to wireless sensor networks is already well advanced, including studies of in-network data aggregation (e.g., [4]–[8]); network connectivity and coverage (e.g., [9], [10]); scheduling (e.g., [11]); clustering (e.g., [12]); and data storage and query processing [13]–[16]. Interest in supporting *spatial* queries using decentralized algorithms and data structures has also begun to grow rapidly, for example in the development of spatial indexes for geosensor networks (e.g., [17]–[21]).

Some spatial operations, like computing the relative neighborhood graph (RNG) or the Gabriel graph (GG), are straightforward to decentralize because they depend, by definition, only on a node's location and that of its one-hop neighbors [22]. More complex spatial queries concern the spatial

M. Duckham is with the Department of Geomatics, University of Melbourne, Victoria 3010, Australia. E-mail: see http://www.duckham.org

D. Nussbaum, J-R Sack, and N. Santoro are with the Department of Computer Science, Carleton University, Ottawa K1S 5B6, Canada

Digital Object Indentifier 10.1109/TC.2010.177

characteristics and relationships between spatially extended objects, like lines, boundaries, regions, or areas. These queries are not so straightforward to decentralize, because the spatial extents of such objects demand coordination amongst spatially dispersed nodes across the network. Existing techniques for satisfying such queries usually aim to take advantage of spatial structures, like Voronoi cells [23], lines [24], boundaries [25]–[27], or the inherent autocorrelation in spatial data [28]–[30].

In this paper, we are directly concerned with queries about the topology of spatial regions, such as "Is region x (topologically) connected?" or "What regions are contained within/adjacent to x?" Recent research has focused on the efficient generation of the geometries of spatial regions, in particular contours or isolines [31]–[34]. However, to date relatively few algorithms have addressed the problem of computing the topology of spatial regions directly (rather than indirectly as a by-product of geometries [35]).

This paper focuses on queries about the current *state* of the region topology (e.g., whether region a contains region b), as opposed to topological *events* (e.g., whether regions a and b "merge" or "split"). Most existing research into DeSC can be classified into either queries about state of the world (e.g., [23], [24], [29], [30]) or queries about geographic events that happen in the world (e.g., [27], [36]–[38]). Events and states are closely related—events can often be inferred from a sequence of states, and the state after (or before) an event can be inferred from the event plus the state before (or after) the event.

Related research has already begun to develop decentralized algorithms for identifying topological *events*, like the appearance, disappearance, merging, and splitting of regions [38]–[40]. However, this previous work does not deal with querying topological states, and indeed often requires initialization with information about the starting topological state (e.g., [38], [41]). Thus, the algorithms in this paper fill a gap in the current research literature, suited to infrequent long-running (queries resident in the network, proactively responding to predefined triggers [42]) or one-off *snapshot* queries about region topology. Further, even where topological changes are expected to occur more frequently (and so event-oriented queries may be more appropriate than our approach) our algorithms still fulfill an important role as an efficient initialization step for long-running queries about topological events.

III. Defining boundary structures

Boundary is a fundamental topological construct in spatial information processing (see for example, the influential "4-intersection model" [43]). This section defines three basic boundary constructs (boundary nodes, boundary cycles, and boundary orientation) in the context of sensor networks.

Wireless sensor networks are conventionally represented as a set of nodes V, connected by a communication network represented as a connected (undirected) graph, $G = (V, E)$. The neighbors of a node $v \in V$ are denoted $nbr(v)$, where $nbr(v) = \{v' | \{v, v'\} \in E\}$. In a geosensor network, the nodes of G are also distributed in space. The (planar) location of the node in geographical space can be represented using a *locator function*, $l : V \to \mathbb{R}^2$.

Each node in a geosensor network can sense information about its immediate (geographical) environment. Formally, we assume a *sensor function* $s : V \to X$, where X is a finite set representing some salient, pairwise disjoint categorization of sensed values (e.g., $X = \{$'hot', 'warm', 'cold'$\}$). Such categories be derived from thresholding scalar fields (e.g., all locations above a certain humidity) or from true binary sensors (e.g., presence or absence of an environmental pollutant). In the simplest case, $X = \{0, 1\}$, a node can sense whether it is "out" or "in" a spatial region.

A natural definition of a boundary node in such a system is as follows (cf. [27], [38]):

Definition 1: A *boundary node* is a node $v \in V$ such that there exists a neighbor $v' \in nbr(v)$ where $s(v) \neq s(v')$.

In other words, a boundary node is a node v that has an immediate 1 hop neighbor v' which senses a different category from v. A "region component" of a network is a connected subgraph where nodes in the subgraph all sense the same category, and neighbors of the region component sense a different category. Formally:

Definition 2: A *piece* of graph $G = (V, E)$ is a set of nodes $V' \subseteq V$ such that for any $v' \in V'$ and $v \in nbr(v')$ then $s(v') = s(v)$ if and only if $v \in V'$. The subgraph $G' \subseteq G$ induced by a piece V' is called a *region component* if a) G' is connected and b) $|V'| > 2$.

Representing the boundary of region components requires the addition of further restrictions on the graph structure. Specifically, assuming the graph G is plane (i.e., is planar along with its planar embedding) it is possible to define the "boundary cycle" of a region component as follows:

Definition 3: A *boundary cycle* of plane region component $G' = (V', E')$ is a simple cycle of a *face* f of G' such that there exists at least one vertex $v \in V - V'$ that is geometrically contained inside f.

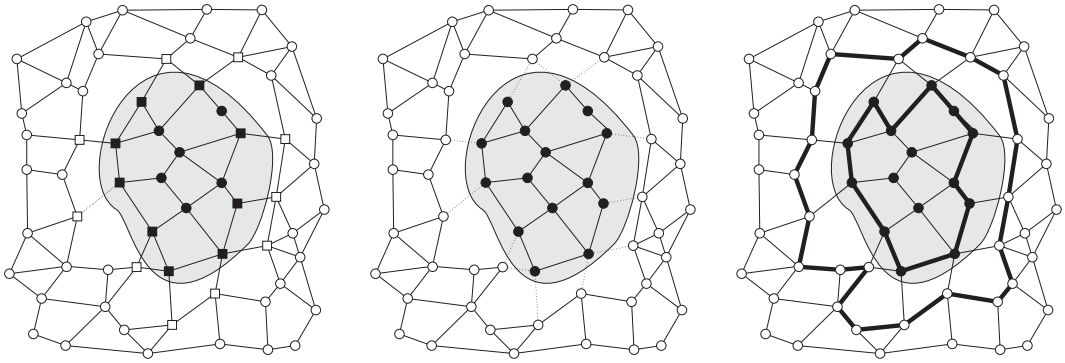
In general, plane graphs may not possess simple cycles for all faces. However, it is a well-known property of 2-connected plane graphs that every face is bounded by a simple cycle [44]. Thus, every 2-connected region component must possess at least one (possibly exterior) boundary cycle.

Figure 1 summarizes diagrammatically the three boundary constructs. In the special case where G is maximally connected (a triangulation), all nodes in the boundary cycle are also boundary nodes. In general plane graphs, every boundary cycle must contain at least one boundary node, but some nodes in the boundary cycle may not be boundary nodes (as illustrated Figure 1c where several non-boundary nodes are part of the boundary cycles).

A. Assumptions

The decentralized computation of boundary structures discussed in the following section relies on four basic assumptions about the graph G and its region components:

1) the graph G is plane;
2) all region components of G are 2-connected;
3) the set of all region component boundary cycles is disjoint, such that any node $v \in V$ can belong to at most one boundary cycle; and



a. Boundary nodes (squares) b. Region components (subgraphs) c. Boundary cycles

Fig. 1. Boundary structures for a graph G with two region components (white and black nodes)

4) all nodes $v \in V$ know their coordinate location.

The first assumption is common for sensor networks, and is fundamental to the definition of the boundary of region components. Many decentralized techniques for constructing planar graphs in a geosensor network have already been devised (e.g., relative neighborhood graph, Gabriel graph, and Delaunay triangulation [22], [45]). The second and third assumptions are required for region components to have simple, non-intersecting boundaries, and have direct analogy to common restrictions in spatial databases and GIS, where polygons are often required to be simple (Jordan) and pairwise disjoint. It is possible to extend our approach to relax assumptions 2 and 3. However, in the interests of clarity and space we treat such extensions as beyond the scope of this paper (but see section VII).

The fourth assumption can potentially be satisfied using conventional positioning systems, like GPS, WiFi, or ultrasonic triangulation. However, in many cases coordinate location provides more information than is strictly required for topological queries. Later sections explore relaxations of this requirement as well as robustness in the face of uncertainty in coordinate location (section VI-D).

IV. Computing boundary structures

Based on the definitions in the previous section, the topological queries explored in the following sections rely on decentralized computation of the three levels of boundary information:

1) *Boundary node detection*: each node locally determines whether it is a boundary node.
2) *Boundary cycle construction*: boundary nodes in a boundary cycle elect a leader to aggregate information about that region component.
3) *Boundary orientation*: the leader for a boundary cycle initiates computation of the orientation of the boundary cycle, representing the polygonal boundary of a region component.

Each level of boundary information requires the construction of the lower level structure (i.e., boundary orientation requires boundary cycle construction, which in turn requires boundary node detection). Depending on the specific topological query, the top level of boundary structure (boundary orientation) may not be required. Adjacency and connectivity queries, for example, can be answered using level 2 boundary cycle information alone (and by implication level 1 boundary node detection). Determining containment relationships between region components requires all three levels.

A. Stage 1: Boundary node detection

Boundary node detection is a straightforward local operation. A node can communicate its sensed value to its immediate 1-hop neighbors, and then compare neighbors' sensed values with its own. A node that detects at least one neighbor with a different sensed value is a boundary node (by definition III).

Boundary node detection requires a total of $O(|V|)$ messages to be sent, and $O(2|E|)$ messages to be received. Note that by Euler's formula, $|E|$ is linearly related to $|V|$ in a plane graph, $|E| \leq 3|V| - 6$. Further, such messages would be normally required by any ad hoc distributed system where nodes need to discover their neighborhood. Since no routing of information is required for boundary node detection (nodes simply broadcast to neighbors), the costs of detecting boundary nodes can arguably be amortized as part of the basic network initialization and infrastructure of a distributed system.

B. Stage 2: Boundary cycle construction

Boundary cycle construction requires nodes to coordinate around the boundary. Intuitively we may construct the boundary cycle using a "boundary tracing" algorithm, that winds around the boundary. In essence, the winding procedure is an adaptation of *face routing* commonly used in wireless sensor networks (e.g., [46], [47]). The key difference is that the "faces" are not (necessarily) faces of the communication graph G; rather they are the faces induced by region components, and reflect the geographic distribution of the underlying sensed values.

Winding around the boundary requires that each node v locally possesses knowledge of the counterclockwise cyclic ordering of its neighbors, represented as the function $c_v : nbr(v) \rightarrow nbr(v)$. The cyclic ordering may be computed locally by each node using basic geometry and knowledge of the coordinate location of neighbors (for example communicated along with their sensed values during stage 1).

For a node $v \in V$ to locally determine the next node in the boundary cycle, it is necessary to consider two cases.

1) if v is a boundary node then $wind_v() \mapsto c_v(v')$ where $s(v') \neq s(v)$ and $s(c_v(v')) = s(v)$
2) otherwise, $wind_v() \mapsto c_v(v')$ where v' is the previous node in the cycle.

Winding around the region component, nodes can route information around the boundary cycle based purely on local information about the sensed values and cyclic ordering of neighbors. Given a local method for routing information around a boundary cycle, constructing the boundary cycle is essentially the well-studied problem of electing a leader in a ring. Many ingenious algorithms for leader election in a ring have been devised (see [48]). In general, leader election problems require that each node possesses a unique identifier. To abstract away from the specific details of the identifier used (for example, possibly using its geographic location), we simply assume each node is able to generate a unique identifier, represented as the function $id : V \to \mathbb{R}$.

The most efficient algorithms for leader election in a ring are $O(n \log n)$, where n is the number of nodes in the ring (the constant hidden by the "big-oh" notation for leader election is small, e.g. 1.44 for bidirectional rings [49]). In the worst case, the number of nodes in the ring (length of the boundary cycle) might approach $|V|$, the number of nodes in the network potentially leading to an overall complexity for leader election of $O(|V| \log |V|)$.

However, the fractal properties of geographic regions mean that the length of region component boundaries is *not* expected to scale linearly with number of nodes. Instead, the length of the boundary cycle for a region component is expected to scale in proportion to $|V|^{0.5*D}$, where $D \in [1, 2)$ is the fractal dimension of the boundary of the region component. For simple Euclidean curves, like the boundaries of rectangles and ellipses, the fractal dimension $D = 1$. Even for more complex geographic features, like lakes and coastlines, the fractal dimension D is typically in the interval $D = 1.2 - -1.3$ [50]. Since for any $D < 2$, $|V|^{0.5*D} = O(\frac{|V|}{\log |V|})$, it follows that leader election is in practice linear in the number of nodes in the network, $O(|V|)$.

C. Stage 3: Boundary orientation

Boundary orientation is a commonly used structure in spatial databases and GIS (for example, as defined in ISO 19107, [51]). A consistent boundary orientation can be deduced by computing the area of the boundary cycle. The area of a polygon is given by the familiar summation $\frac{1}{2} \sum_{i=1}^{n} x_i y_{i+1} - x_{i+1} y_i$, where the sequence of coordinates in the polygon is $\langle (x_1, y_1), (x_2, y_2), ..., (x_n, y_n) \rangle$ and $(x_1, y_1) = (x_n, y_n)$. Since at each step the area function only requires information about the coordinate location of pairs of neighboring nodes, area is straightforward to compute in decentralized way in the boundary cycle. Beginning with leader of the boundary cycle, elected in stage 2 above, the partial sum can be passed around the cycle, each node computing the next term in the summation, until it arrives back at the leader [52].

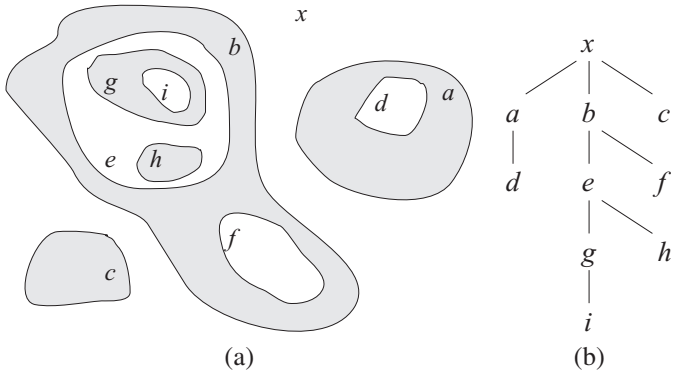


Fig. 2. Areal object (a) and corresponding topology represented as a containment tree (b) after [53]

Following leader election, operations requiring coordination around the boundary will be linearly related to the boundary length, $|V|^{0.5*D}$. Computing boundary orientation involves two complete traversals of boundary cycles (one to compute area, one to inform all boundary nodes of the orientation), leading to a complexity of $O(|V|^{0.5*D})$.

D. Overall communication complexity

It follows from the discussion above that the communication complexity of an optimal algorithm to construct all three levels of boundary structure is linear in the number of nodes in the network, $O(|V|)$, since all the stages (boundary node detection, leader election, boundary cycle construction, and boundary orientation detection) have at worst linear communication complexity.

V. Decentralized algorithms for topological queries

The boundary structures described above form the basis for a range of more sophisticated topological queries. This section presents a family of decentralized algorithms (called IN-TORQUE, In-Network TOpological Region QUEries) that demonstrates how the basic boundary structures can be used to respond to fundamental topological queries, including containment and adjacency queries.

A. Containment queries: Topology of areal objects

The first variant of the IN-TORQUE algorithm determines the topology of a complex *areal object*, comprising a finite number of regions (including holes and islands). The topology of an areal object can be described as a tree, with its regions ordered by containment, after [53]. Figure 2 shows an example of an areal object comprising many regions (and holes and islands), and the corresponding containment tree representation of the topology of the areal object.

The objective of the first variant of the IN-TORQUE algorithm given in Algorithm 1 is to generate in the network the containment tree for a complex areal object, ultimately reported back to some known sink node. Algorithms 1 and 2 adopt the analysis and presentation style of [48], with nodes transitioning between four states $\{\text{INIT}, \text{IDLE}, \text{BNDY}, \text{LEAD}\}$ as summarized in Figure 3.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.

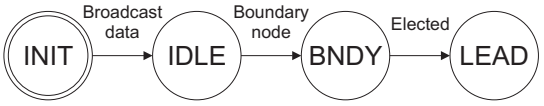


Fig. 3. States and transitions for Algorithms 1 and 2

For the purposes of this example, it is assumed that 1) there exists one sink node contained in the *exterior* face of the areal object (i.e., in the root component of the containment tree), and that 2) all network nodes can route information to this sink (for example using geographic routing, e.g., [46]). To abstract away from the details of the protocol used to route information back to the sink, Algorithm 1 assumes the existence of a local function $rte_v : V \rightarrow nbr(v)$, such that for a source node $v \in V$ routing a message to a sink node $sid \in V$, the next node on the path is given by $rte_v(sid)$. It is straightforward to extend the algorithm to more sophisticated reporting scenarios, such as reporting to multiple different sink nodes, retaining the generated topological information within the network at the region component boundaries, or where nodes do not possess knowledge of an exterior sink node, although these possibilities are not considered further here.

The IN-TORQUE $_{\text{areal}}$ protocol (Algorithm 1) is in effect a decentralized version of the well-known *semi-line algorithm*, commonly used for point-in-polygon tests in GIS and spatial databases [54]. The centralized semi-line algorithm determines containment using computational geometry to count the number of intersections of a semi-line with polygon boundary. By contrast, the decentralized IN-TORQUE $_{\text{areal}}$ algorithm routes a message from each boundary cycle leader to the exterior, updating the message at each boundary node encountered with information about the boundary cycle crossed.

Algorithm 1 operates by first constructing all three levels of boundary structure, boundary nodes, cycles, and orientation. Boundary orientation (section IV-C) is required for this computation in order to distinguish an outer boundary (of a region) or an inner boundary (of a hole). In Algorithm 1, the local variable *hol* is used to store the information generated about boundary orientation. Since messages are routed from a leader to the first boundary cycle node anticlockwise from the exterior, *hol*=0 indicates a positive area for the region, and so a outer boundary of a region; conversely *hol*=1 indicates a negative area for the region, and so an inner boundary of a hole (see Figure 4a).

A message is then routed from the elected leader for each region component to the sink. The leader for each boundary cycle creates a "report" message containing the leader id and an initial "score" for the message. The initial score depends on the boundary orientation and on the state of the node receiving the first hop. A message where the first hop takes it from an inner (hole) boundary into the hole, or from an outer (region) boundary into the region, are initialized with score -2; otherwise messages are initialized with score -1 (Figure 4b).

The report message is then forwarded toward the sink. A message crossing *into* an inner (hole) boundary or *out of* an outer (region) boundary will have its score incremented by 1. Conversely, crossing *out of* an inner (hole) boundary or *into* an outer (region) boundary causes the message score to be decremented by one. When the score is incremented to zero, the containing boundary has been detected (Figure 4c). At this point the identity of the containing boundary cycle is added to the message, and the message is subsequently forwarded unaltered to the sink. When the sink has received messages from all boundary cycles, the entire topology tree can be computed from the partial order induced by all containment relations (with the addition of the exterior as the parent of any messages that have encountered no containing regions).

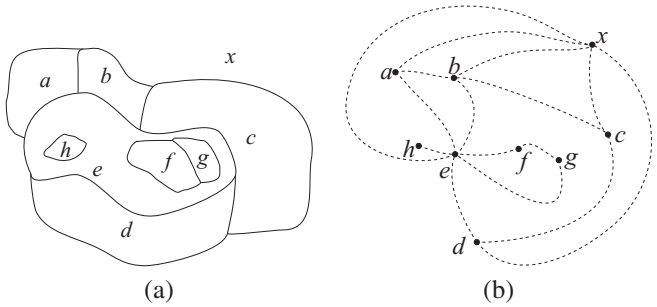


Fig. 5. Set of geographic region boundaries (a), along with adjacency graph for regions (b)

In addition to the basic $O(|V|)$ communication complexity of the IN-TORQUE algorithm discussed in section IV-D, the IN-TORQUE $_{\text{areal}}$ protocol requires additional $O(m)$ time to complete, where m is the length (number of nodes) in the paths from each boundary cycle leader to the sink node. This length will depend on both the number of boundary cycles and the routing protocol used. However, for a non-fractal Jordan path through the network, this is expected to scale $O(\sqrt{|V|})$, as discussed previously.

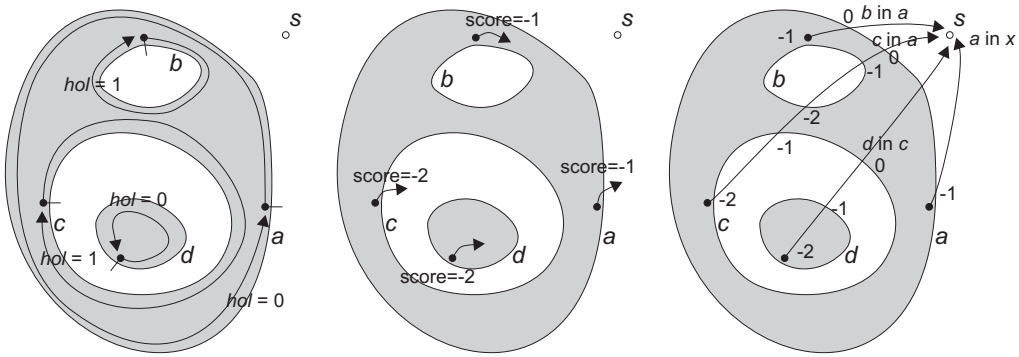
B. Adjacency queries: Topology of sets of region components

A second fundamental query for centralized spatial computing is determining adjacency between different regions. Adjacency here is taken to mean any two boundaries that directly abut one another (whether or not one is the boundary of a containing region). Figure 5 shows an example of a set of region boundaries and their adjacency relationships.

Computing adjacency in the network requires only levels 1 and 2 boundary structures (nodes and cycles). Algorithm 2 presents the adjacency variant of the IN-TORQUE algorithm. As expected, much of the algorithm is common to Algorithm 1, specifically the construction of boundary nodes and boundary cycles.

In Algorithm 2 during the construction of the boundary, each boundary node stores information about which neighbors are also boundary nodes, represented for each node v as the function $bids_v : \emptyset \rightarrow 2^{\mathbb{Z}}$. As nodes construct a boundary cycle, the identity of the boundary cycle leader is communicated to and stored by boundary node neighbors, represented for each node v as the function $bnbr_v : \emptyset \rightarrow 2^{nbr(v)}$.

After boundary cycle construction, a message passed around the boundary cycle by the region component leader is used to collect the identities of all adjacency boundary cycles. This information can be efficiently collated without duplication by



a. Boundary orientation b. Initial score c. Updating scores

Fig. 4. Report message scoring for boundary cycles a, b, c, and d, with leaders (black nodes) reporting to sink s

each region component leader by simply comparing boundary cycle identities (e.g., only report smaller boundary identities), and subsequently forwarding this information to the sink. As for the IN-TORQUE $_{\text{areal}}$ protocol, once at the sink the combination of this qualitative adjacency information constitutes the adjacency map for the entire sensed area.

As for the IN-TORQUE $_{\text{areal}}$ protocol, the overall communication complexity of the IN-TORQUE $_{\text{adjacency}}$ protocol is $O(|V|)$.

C. Further queries

Containment and adjacency represent basic topological queries common to any spatial computing system. However, a range of other queries may be similarly satisfied. For example, while containment and adjacency have been presented here as separate algorithms, both rely on the same basic boundary structures and can easily be combined into a single $O(|V|)$ protocol. A combined algorithm can determine containment and adjacency at the same time, effectively constructing a "topological map" for the areal object.

Connectivity queries (such as "Is the bushfire front connected?"), are also simple to satisfy using any of the structures already discussed. Modifying either of the algorithms above slightly to additionally communicate the sensed values (and not simply the region identity) back to the sink, would allow the sink to search through the region identity–sensed value pairs. The area covered by a particular sensed value will be disconnected if and only if that sensed value that appears more than once with two different region identities.

Extending the scenario slightly further to allow nodes to sense environmental variables from more than one domain enables further queries to be satisfied using the same boundary approach. In this category, *overlay* is another basic function of any GIS or spatial database which constructs all the non-empty intersections of two or more sets of polygonal regions. Modifying the algorithms above to compute the overlay of more than one sensed data domain is trivial. For example, assuming n sensed data domains and associated sensor functions, $s_1 : V \to X_1$, $s_2 : V \to X_2$, ..., $s_n : V \to X_n$, the topological map for the overlay of the sensed domains $X_1, ..., X_n$ can be computed by replacing the function $s : V \to X$ in Algorithms 1 or 2 with a combined function $s : V \to X_1 \times X_2 \times ... \times X_n$ such that $s(v) \mapsto (s_1(v), s_2(v), ..., s_n(v))$.

VI. Empirical analysis

In addition to the computational analysis in previous sections, the performance of the IN-TORQUE algorithm was tested experimentally using simulation. The experiments empirically investigated four key features of the algorithm: scalability, load balancing, latency, and robustness. The experiments were conducted on the IN-TORQUE $_{\text{areal}}$ algorithm, as this was the most sophisticated algorithm requiring all three levels of boundary information (section IV).

A. Scalability

A scalability experiment was conducted on simulated networks varying in size from 1000 to 20000 nodes. Node location in the networks was randomized. The UDG (unit distance graph) was generated for each network (i.e., all nodes closer than a communication distance c were connected by one-hop communication). To ensure comparable levels of connectivity for all networks across the range of different network sizes, the communication distance was set relative to the node density, $\frac{A}{|N|}$. Specifically, the distance c was chosen to be $\sqrt{4\frac{A}{|N|}}$, where N is the set of nodes in the network and A is the area over which these nodes were randomly distributed. This communication distance was chosen to be large enough to ensure that the network was connected, and small enough to ensure a reasonably sparse communication graph (to model a network distributed over geographic space).

As a control, a brute force ("sense-and-transmit") algorithm was used to forward each node's data along the shortest paths tree [4] to a sink node located at the center of the network (the best case in terms of total paths length) for subsequent centralized processing. For the decentralized IN-TORQUE algorithm, the network was structured as a locally computed planar subgraph of the UDG (the relative neighborhood graph [55]), with messages georouted to the sink [46]. To guarantee that the sink was in the exterior of the monitored areal object (cf. section V-A), the sink was located at one extreme of the network for experiments on the IN-TORQUE algorithm (the worst case in terms of total paths length).

Figure 6 compares the results of the experiments on the scalability of the IN-TORQUE $_{\text{areal}}$ and brute force "sense-and-transmit" algorithms. The experiments were conducted on a

Algorithm 1 IN-TORQUE $_{\text{areal}}$ protocol for node v

States: $\mathcal{S} = \{$INIT, IDLE, BNDY, LEAD$\}$
Local variables: *rid*, *hol*
Local knowledge: *sid* (sink id)

INIT
  Spontaneously
    Broadcast('INIT', $l(v)$, $s(v)$) to *nbr*(v)
    Become IDLE

IDLE
  Receiving('INIT',l',s') from v'
    Store $l(v') \mapsto l'$ and $s(v') \mapsto s'$
    Construct c_v : *nbr*$(v) \rightarrow$ *nbr*(v)
    if $s' \neq s(v)$ **then**
      Become BNDY
      Send('LEAD', *id*(v)) to *wind*$_v()$
  Receiving('LEAD', i) from v'
    Send('LEAD', i) to $c_v(v')$
  Receiving('AREA', a) from v'
    $a \leftarrow a + l(v).x * l(v').y - l(v).y * l(v').x$
    Send('AREA', a) to $c_v(v')$
  Receiving('RING', i, s) from v'
    Send('RING', i, s) to $c_v(v')$
  Receiving('RPRT', c, i, s) from v'
    Send('RPRT', c, i, s) to *rte*$_v(s)$

BNDY
  Receiving('LEAD', i) from v'
    if $i =$ *id*(v) **then**
      Become LEAD
      Send('AREA', 0) to *wind*$_v()$
    else
      if ⟨election condition met⟩ **then**
        Send('LEAD', i) to *wind*$_v()$
  Receiving('AREA', a) from v'
    $a \leftarrow a + l(v).x * l(v').y - l(v).y * l(v').x$
    Send('AREA', a) to *wind*$_v()$
  Receiving('RING', i, s) from v'
    Store *hol* $\leftarrow s$
    Store *rid* $\leftarrow i$
    Send('RING', i, s) to *wind*$_v()$

LEAD
  Receiving('AREA', a) from v'
    $a \leftarrow a + l(v).x * l(v').y - l(v).y * l(v').x$
    Store *hol* $\leftarrow 0$
    if *sign*(a) is +ve **then**
      Store *hol* $\leftarrow 1$
    Store *rid* $\leftarrow i$
    Send('RING', *id*(v), *hol*) to *wind*$_v()$
  Receiving('RING', i, s) from v'
    if *hol* $= s($*rte*$_v($*sid*$))$ **then**
      Send('RPRT', -1, *rid*) to *rte*$_v($*sid*$)$
    else
      Send('RPRT', -2, *rid*) to *rte*$_v($*sid*$)$

BNDY, LEAD
  Receiving('RPRT', c, i) from v'
    Wait until *rid* is set
    if $s(v') \neq s(v)$ **then**
      $c \leftarrow c - 1 + 2 * hol$
    if $s($*rte*$_v($*sid*$)) \neq s(v)$ **then**
      $c \leftarrow c + 1 - 2 * hol$
    if $c = 0$ **then**
      Send('DONE', i, *rid*) to *rte*$_v($*sid*$)$
    else
      Send('RPRT', c, i) to *rte*$_v($*sid*$)$

ANY
  Receiving('DONE', i_1, i_2) from v'
    Send('DONE', i_1, i_2) to *rte*$_v($*sid*$)$

Algorithm 2 IN-TORQUE $_{\text{adjacency}}$ protocol for node v

States: $\mathcal{S} = \{$INIT, IDLE, BNDY, LEAD$\}$
Local variables: *rid*
Local knowledge: *sid*

INIT
  Spontaneously
    Broadcast('INIT', $l(v)$, $s(v)$) to *nbr*(v)
    Become IDLE

IDLE
  Receiving('INIT', l', s') from v'
    Store $s(v') \mapsto s'$
    Construct c_v : *nbr*$(v) \rightarrow$ *nbr*(v)
    if $s' \neq s(v)$ **then**
      Become BNDY
      Store *bnbr*$_v() \mapsto$ *bnbr*$_v() \cup \{v'\}$
      Send('LEAD',*id*(v)) to *wind*$_v()$
  Receiving('LEAD', i) from v'
    Send('LEAD', i) to $c_v(v')$
  Receiving('RING', i) from v'
    Send('RING', i) to $c_v(v')$
  Receiving('ADJY', I) from v'
    Send('ADJY', I) to $c_v(v')$

BNDY
  Receiving('LEAD', i) from v'
    if $i =$ *id*(v) **then**
      Become LEAD
      Broadcast('BIDS', i) to *nbr*(v)
      Send('RING', i) to *wind*$_v()$
    if ⟨election condition met⟩ **then**
      Send('LEAD',i) to *wind*$_v()$
  Receiving('RING', i, s) from v'
    Store *rid* $\leftarrow i$
    Broadcast('BIDS', i) to *nbr*(v)
    Send('RING', i) to *wind*$_v()$
  Receiving('ADJY', I) from v'
    Wait until 'BIDS' message received from all nodes in *bnbr*$_v()$
    Send('ADJY', $I \cup$ *bids*$_v()$)

LEAD
  Receiving('RING', i, s) from v'
    Wait until 'BIDS' message received from all nodes in *bnbr*$_v()$
    Send('ADJY', *bids*$_v()$) to *wind*$_v()$
  Receiving('BIDS', I) from v'
    if $I \neq \emptyset$ **then**
      Send('DONE',i,I,s) to *rte*$_v(s)$

BNDY, LEAD
  Receiving('BIDS', i) from v'
    if $i <$ *rid* **then**
      Store *bids*$_v() \mapsto$ *bids*$_v() \cup \{i\}$

ANY
  Receiving('DONE', i, I) from v'
    Send('DONE', i, I) to *rte*$_v(s)$

simple, arbitrarily generated areal object, containing one outer region, two holes, one of which contained two island regions. Using the same areal object across all experiments ensured comparability between experiment sets. Further, experimenting on a relatively simple areal object was necessary to ensure the topological relationships could be discerned at all the spatial granularities tested (i.e., even at networks with just 1000 nodes). However, the algorithm was also verified to operate correctly with a wide variety of areal objects of arbitrary complexity at a range of different spatial granularities.

Scalability was measured in terms of number of messages

Process	Factor a	Power b	R^2
Total "sense-and-transmit"	0.274	1.486	1.000
Total IN-TORQUE $_{\text{areal}}$	9.911	0.807	0.999
LEAD	14.11	0.628	0.999
INIT	1	1	1.000
RING	7.342	0.519	0.999
AREA	7.371	0.519	0.999
RPRT/DONE	2.511	0.534	0.995

TABLE I
RESULTS OF POWER REGRESSION ANALYSIS ($y = a.x^b$) FOR RESPONSE CURVES IN FIGURE 6

(sent) across the range of network sizes. Figure 6 shows the total number of messages required for each algorithm averaged over 25 randomized simulations. The figure also shows the component messages required for each stage of the IN-TORQUE algorithm (associated with LEAD, INIT, RING, AREA, and RPRT/DONE messages). Figure 6 is presented as a log-log graph, since the numbers messages and nodes vary over several orders of magnitude (from tens to millions of messages, from thousands to tens of thousands of nodes).

As expected, the INIT (boundary node detection) stage exhibits linear communication complexity with increasing network size. The LEAD stage (leader election) uses the MinMax algorithm for leader election in a unidirectional ring [48]. This algorithm has $1.44n \log n + O(n)$ complexity, where n is the length of the region boundary (hence, as discussed above n scales as a function of $\sqrt{|V|}$). Also as expected, the AREA (compute area), RING (store orientation), and RPRT/DONE (report to sink) message components, which are approximately linear in the length of boundary cycle, make up a relatively small components of the overall communication cost.

The results of a regression analysis for the different response curves in figure 6 are shown in Table I. A power regression was used, fitting a curve of the form $y = a.x^b$ to each of the response curves in Figure 6. All the regression curves achieved very high goodness of fit (varying from R^2 values of 0.995 to 1.000). The most notable feature of these results is that they support the intuitive interpretation of the results above, with the INIT being of order $O(|V|)$, and with LEAD, RING, AREA, and RPRT/DONE components being of order $O(|V|^b)$ where $0.5 < b < 0.63$. Overall, the total IN-TORQUE $_{\text{areal}}$ scalability improves on the theoretical worst case $O(|V|)$ (approximately $O(|V|^{0.81})$). By contrast, the centralized sense-and-transmit approach is significantly less scalable, exhibiting scalability of $O(|V|^{1.49})$, albeit with a small constant factor (≈ 0.3).

By making some assumptions about the length of messages sent, it is also possible to compare the total amount of data transmitted using the two approaches. For the IN-TORQUE $_{\text{areal}}$ algorithm, we require: 33 bits for INIT messages (2×16 bit x and y coordinates plus 1 bit sensed value, in or out of the region); 16 bits for LEAD messages (16 bit unique leader id); 48 bits for AREA messages (32 bit x, y coordinate plus 16 bit partial area sum); 1 bit for RING messages (clockwise or anticlockwise boundary orientation); and up to 32 bits for RPRT/DONE messages (2×16 bit leader ids for contained and containing regions).

For the brute force approach, each node must send 33 bits to the sink (32 bit x, y coordinate and 1 bit sensed value). The brute force approach might additionally send each 16 bit unique node id to the sink, but we assume for efficiency the x, y coordinate can also provide a unique id (and is required anyway for centralized computation of region boundary and topology). Potentially, the sensed value may be aggregated in the network. Again for the most stringent test, we assume the optimal (but highly unlikely) case of perfect aggregation, where the sensed value accounts for only $|V|$ bits total transmission (i.e., where effectively each sensed value is transmitted once, and then aggregated at the parent node in the shortest path tree). However, each x, y coordinate is unique. Even the location of a single node can potentially influence the centralized computation of containment relationships. Consequently, no data aggregation is possible for the coordinates; each must be retransmitted in full through to the sink for topological queries to be reliably satisfied.

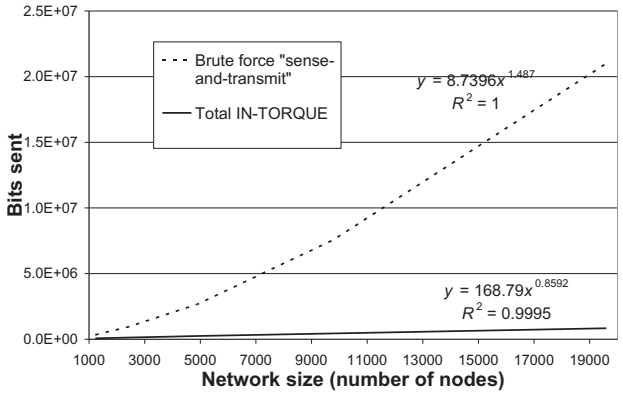


Fig. 7. Scalability of IN-TORQUE $_{\text{areal}}$ versus centralized "sense-and-transmit" algorithms, in terms of numbers of bits sent

Figure 7 compares the scalability in terms of bits transmitted for the two algorithms. In addition to the response curves, Figure 7 also shows the results of a regression analysis similar to that in Table I. For message length, the IN-TORQUE $_{\text{areal}}$ exhibiting scalability of order $O(|V|^{0.86})$, again comparing well with the brute force centralized algorithm, order $O(|V|^{1.49})$. In cases where the initialization costs can be treated as preprocessing required for the establishment of network services, the scalability of bits transmitted for the IN-TORQUE $_{\text{areal}}$ improved even further, to $O(|V|^{0.6})$.

Discussion: The experiments provide strong evidence that the decentralized IN-TORQUE algorithm is highly scalable in terms of both total numbers of messages and bits transmitted. On average, our algorithm required approximately 2.5 messages per node for the smallest network, scaling to about 1.5 messages per node for the largest network. By comparison, the brute-force "sense-and-transmit" approach required an average of more than 8 messages per node for the smallest network, rising to 33 messages per node for the largest network.

As might be expected, in our experiments the scalability of the algorithm significantly improves on the theoretical worst case $O(|V|)$. The INIT component dominates this overall scalability. As argued previously, algorithm initialization, where every node broadcasts its sensed value to its immediate 1-

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.

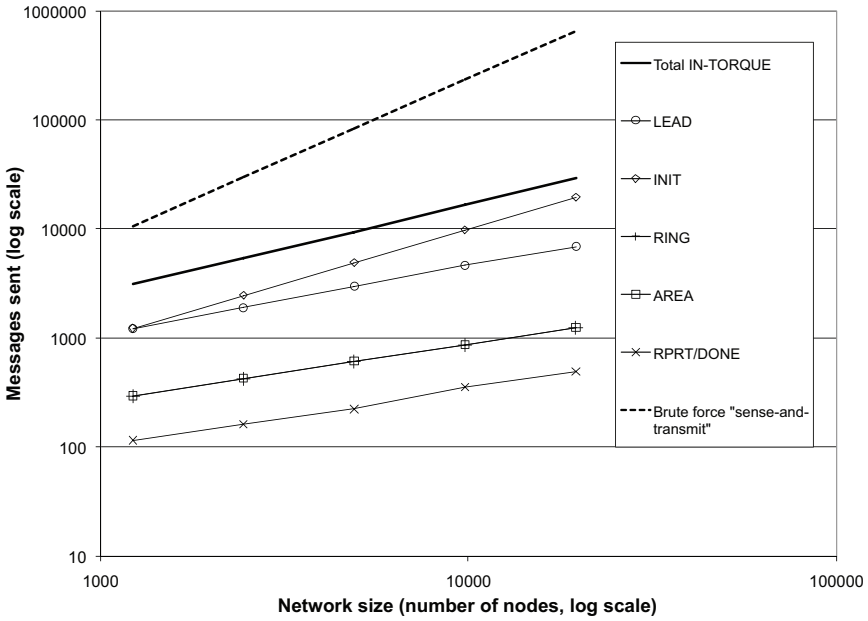


Fig. 6. Scalability of IN-TORQUE $_{\text{areal}}$ versus centralized "sense-and-transmit" algorithms, in terms of numbers of messages sent

hop neighbors, could potentially be amortized in the costs of standard network initialization. In our experiments, the algorithm achieved $O(|V|^{0.6})$ scalability for both total number of messages and total bits transmitted if initialization costs were removed. Indeed, while we have included the initialization costs in our evaluation of the IN-TORQUE algorithm in the previous section, we have *ignored* similar initialization costs required by the brute force centralized approach in setting up the shortest paths routing tree. Were these costs included it would further increase the gap in efficiency between the centralized approach and the IN-TORQUE algorithm.

B. Load balancing

Arguably more important than the overall scalability is the load balancing characteristics of the IN-TORQUE algorithm. Poor load balancing will lead to more rapid depletion of energy resources at some nodes, potentially resulting in disconnected networks and low suitability for long-term monitoring. Figure 8 compares the load balance of the IN-TORQUE $_{\text{areal}}$ algorithm with a sense-and-transmit approach (for a specific medium-sized simulation of 4900 nodes). The figure shows a typical histogram of the number of messages per "edge" (i.e., between two neighboring nodes). The IN-TORQUE $_{\text{areal}}$ algorithm in Figure 8a exhibits dramatic improvements in load balancing over the simulations of the same scenario using the sense-and-transmit approach in Figure 8b. Using the IN-TORQUE $_{\text{areal}}$ algorithm the load on most transmission links is just two messages, with no pair of nodes exchanging more than 22 messages (i.e., within the 16–32 messages class in the histogram in Figure 8a). While the modal number of messages per edge using the sense-and-transmit approach is just 1, most pairs of nodes exchange many more messages. More than 150 edges (approximately 3% of the node pairs in the network) bear loads of above 100 messages, with a handful of nodes exchanging almost 1000 messages.

Discussion: The experimental results strongly indicate the relative efficiency gains of the IN-TORQUE when compared with a centralized sense-and-transmit algorithm. In the case of the network in Figure 8, some nodes transmit almost 50 times more messages using the centralized approach when compared with the IN-TORQUE algorithm. Larger networks are only expected to increase this substantial load imbalance.

C. Latency

A third important computational characteristic to consider is message latency, for example measured as the largest number of hops of any message required by the algorithm. In the case of the brute force sense-and-transmit approach, the maximum number of hops of any message is expected to equal the diameter of the network (assuming the worst case, where the sink is at an extreme edge of the network). For a network of $|N|$ nodes randomly located in the plane, diameter is expected to scale $O(\sqrt{|N|})$.

By contrast, the maximum length of any sequence of messages in the IN-TORQUE algorithm is expected to depend on the length of the region boundary, and distance from boundary leader to the sink node. In general, the largest number of hops for the the IN-TORQUE algorithm will result from: 1 hop (INIT message), plus $3m$ hops (LEAD, RING, and AREA messages, where m is the length of the boundary of the region), plus n hops (RPRT/DONE message, where n is the length of the path from the boundary leader to the sink). As already discussed, the number of hops of cycles around the boundary and paths to the sink are expected to scale approximately $O(\sqrt{|N|})$. Thus as for the brute-force approach, the latency of the IN-TORQUE algorithm is exhibit comparable scalability, approximately $O(\sqrt{|N|})$.

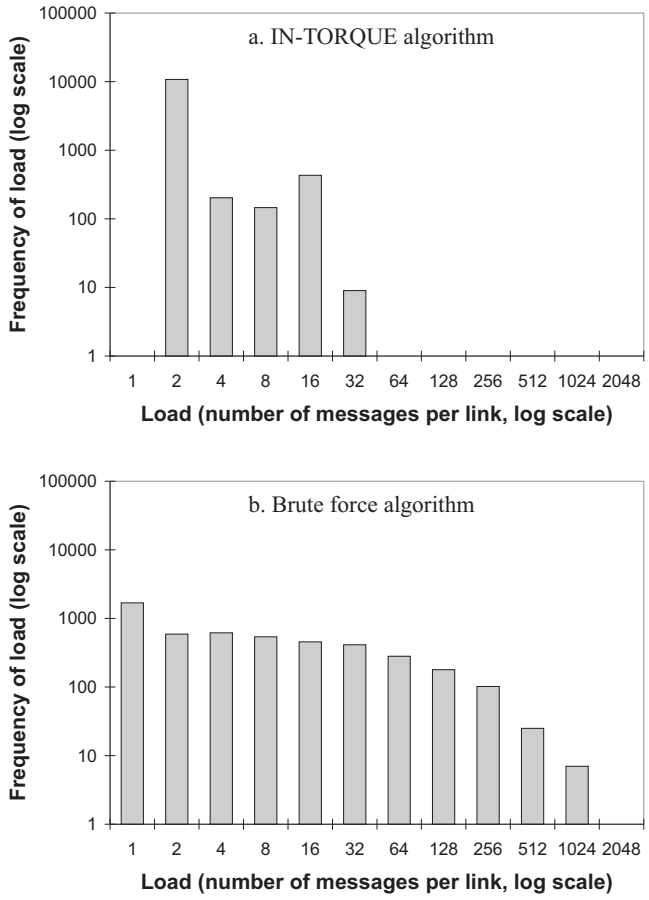


Fig. 8. Load balancing in terms of number of messages per edge for (a) IN-TORQUE $_{areal}$ algorithm; and (b) brute force sense-and-transmit algorithm (for medium-sized network, 4900 nodes)

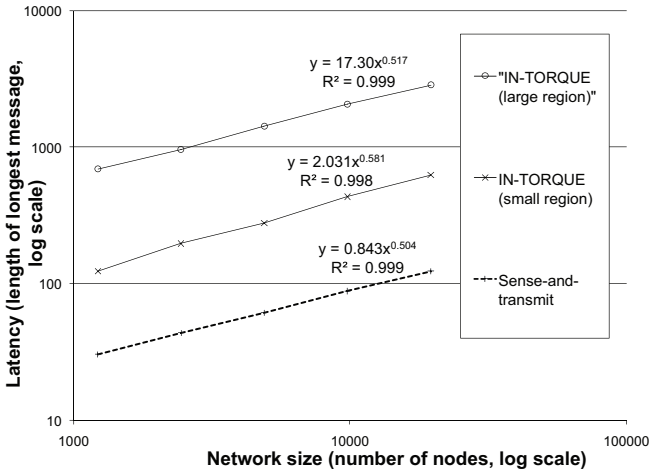


Fig. 9. Latency (in terms of maximum length of any sequence of messages) for NONAME$_{areal}$ algorithm and brute force sense-and-transmit algorithm

A series of further experiments, which measured the length of the longest sequence of messages across different network sizes, investigated these expectations. Figure 9 summarizes the results. The experiments confirmed that the latency of the IN-TORQUE algorithm has approximately $O(\sqrt{|N|})$ scalability, with a constant factor that depends strongly dependent on the size of the region component being monitored. For example, a power regression on the message lengths associated with the smallest region from the earlier experiments indicated a latency of $2.031|N|^{0.581}$ ($R^2 = 0.998$) across all the network sizes tested (i.e., from $|N|$=1000 to 20000 nodes). A similar regression on latency of messages associated with the largest region used in the earlier experiments revealed a scalability of $17.3|N|^{0.517}$ ($R^2 = 0.999$).

By contrast, the latency for the brute force sense-and-transmit approach exhibited comparable order of scalability to the IN-TORQUE algorithm, but with lower constant factors. A power regression on the longest messages resulting from the sense and transmit approach yielded a scalability of $0.843|N|^{0.504}$ ($R^2 = 0.999$, see Figure 9).

Discussion: It is entirely possible for a small enough region, close enough to the sink, to result in a *lower* latency using the IN-TORQUE algorithm when compared with the brute force approach. For example, a small region with 20 nodes in the boundary cycle, and a leader 5 hops from the sink will result in a longest message length for the IN-TORQUE of 1+3×20+5=66 hops. If this region occurs in a large network (for example the largest networks used in our experiments with 20,000 nodes and diameter of around 120 nodes) then the IN-TORQUE algorithm is expected to exhibit lower latencies than the sense and transmit approach.

However, in the example of our experiments, the IN-TORQUE algorithm exhibited higher latencies than the sense-and-transmit approach. The latency associated with the IN-TORQUE algorithm is strongly dependent on the size of the largest region being monitored, as well as the location of the sink relative to the region. Thus, in general, it is to be expected that the IN-TORQUE algorithm will be associated with higher latencies than a brute force, sense-and-transmit approach, except in exceptional cases of monitoring only small regions. Although this increased latency is a cost of using the IN-TORQUE algorithm, the latencies associated with the IN-TORQUE algorithm remain of comparable orders of scalability to the sense-and-transmit approach (i.e., approximately $O(\sqrt{|N|})$), differing only by a constant factor (in our experiments between about 2 and 20 times greater latencies for the IN-TORQUE algorithm).

D. Robustness

Finally, the experiments also investigated the robustness of the IN-TORQUE $_{areal}$ algorithm, in terms of its tolerance of uncertainty of location. Thus far, simulations have assumed that nodes have access to precise and accurate information about their own location. In actuality, such information is likely to be highly resource-intensive or even impossible to obtain. Tolerance to imperfect location information is one of the key motivations for monitoring the *qualitative* spatial properties of phenomena, like the topological properties of complex areal objects, rather than the quantitative characteristics, like the geometry of the boundaries of areal objects.

Figure 10 presents the results of experiments on the robustness to positional inaccuracy of the IN-TORQUE $_{areal}$ algorithm.

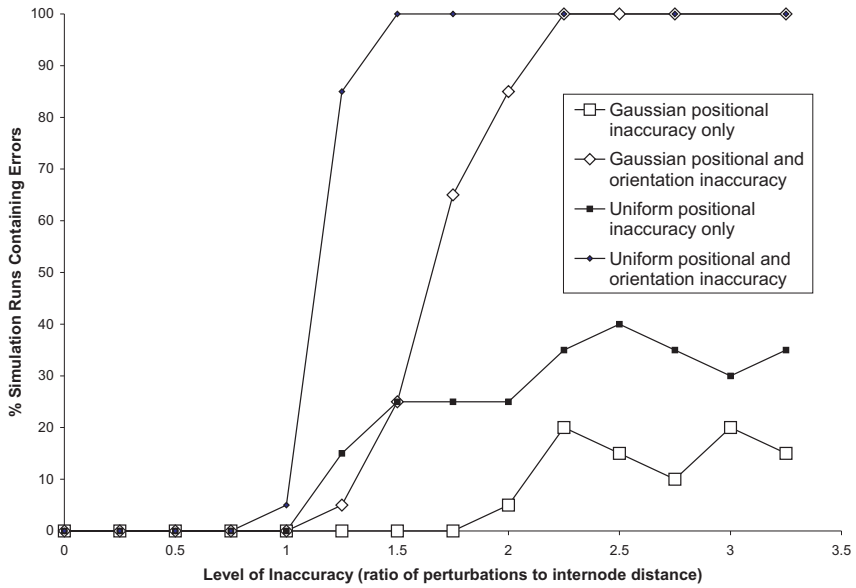


Fig. 10. Robustness of IN-TORQUE $_{areal}$ algorithm to positional and orientation inaccuracy

The experiments were based on simulations where every node's knowledge of their location was randomly perturbed in the x and y directions. Perturbations were selected first from a uniform random probability distribution with maximum perturbation p, and then repeated for perturbations drawn from a Gaussian probability distribution, with standard deviation σ. Unlike previous experiments, the robustness simulations were conducted using a regular, grid-based network, to avoid randomly perturbing an already randomized network. To enable comparison between results, the overall level of inaccuracy is measured as the ratio between the maximum perturbation p (uniform probability distribution) or standard deviation σ (Gaussian distribution) and the internode distance d (distance between a node and its one-hop neighbors). The internode distance d is fixed by the use of a regular grid of nodes, connected using a "rook's case" network. For example, for a network with an internode distance of d=20 units and maximum perturbation p=15 units (uniform distribution) or standard deviation σ=15 units (Gaussian distribution), the level of inaccuracy would be 15/20=0.75. Increasing node perturbation increases the measure of node inaccuracy (e.g., p or σ=30 units, d=20 units then the level of inaccuracy would be p/d=σ/d=30/20=1.5).

The robustness of the algorithm was measured as the percentage of simulations out of 20 repetitions (for each inaccuracy level) that resulted in anything other than a perfect detection of the topology of complex areal object. The magnitude of the perturbation, shown along the x axis of Figure 10, was varied across a series of experiments, with the resulting percentage robustness measured and displayed on the y axis of Figure 10.

Further, Figure 10 shows the results of two different sets of simulations. In the first experiment, perturbations were used to alter the node's knowledge of its coordinate location, but the qualitative orientation of neighbors around the node was assumed to be accurate. These experiments model the situation in networks where qualitative orientation is found independently of node location (for example, using ultrasound or RF direction-finding sensors). In the second experiment, node orientation was derived from node position, so increasing positional inaccuracy also led to increasingly inaccurate knowledge of the relative cyclic ordering of the neighbors around each node. These experiments model the situation where qualitative orientation is derived from quantitative positioning systems (like RF triangulation or GPS).

Overall, the IN-TORQUE algorithm exhibits relatively high levels of robustness to inaccuracy. For example, below a node inaccuracy level of 0.75, the algorithm still correctly identifies the topological structure of the complex areal object in all cases. For a uniform distribution above a node inaccuracy level of 1.0 (i.e., where the maximum perturbation is comparable to or greater than the internode distance) the algorithm does begin break down, becoming unreliable. Using a Gaussian distribution, the algorithm performs even more robustly, and in the case of no orientation inaccuracy, the algorithm still performs perfectly up to a level of inaccuracy of 1.75 (i.e., where the standard deviation of perturbations is 1.75 times the internode distance). In general, experiments using Gaussian perturbation outperform those drawn from a uniform distribution. This result can be interpreted as indicating the algorithm's robustness to infrequent, large outliers: in the Gaussian distribution a minority (32%) of the perturbations will be larger than the maximum possible perturbation using a uniform distribution. Conversely, in the Gaussian distribution, a larger proportion of perturbations will be closer to the mean than in the uniform distribution. As might be expected, for simulations where inaccuracy affects only node position, much lower levels of unreliability result than for simulations where inaccuracy in both position and orientation are modeled.

Discussion: Inaccuracy and imprecision are endemic features of any positioning system, especially in resource-limited geosensor networks. The experiments demonstrate that the IN-TORQUE algorithm can tolerate considerable positional inaccuracy in nodes. Such robustness is an important advantage of using qualitative algorithms for computing with spatial information. The errors observed in the simulations of positional inaccuracy only were all topological, where the topological relationship for one or more regions of the complex areal object were incorrectly detected. These errors arose where the magnitude of perturbations increases to such an extent that the sign of the area was inverted, and so the topological relationship between regions was incorrect. Simulations that included both positional and orientation accuracy also result in algorithm failure, primarily caused by the failure of the underlying face-routing in the presence of very high positional inaccuracy.

VII. Conclusions

This paper has demonstrated how spatial queries about the topology of spatial regions, monitored by a wireless sensor network, can be satisfied using in-network, decentralized algorithms. The approach is founded on the decentralized construction of fundamental boundary structures: boundary nodes, boundary cycles, and boundary orientation. The family of IN-TORQUE algorithms can detect containment and adjacency between regions, as well as providing a basis for other topological queries such as connectivity and overlay. The approach is shown both analytically and empirically to be efficient, scalable, and robust, although at the cost of somewhat increased message latency. As such, this research contributes to broader efforts to construct the foundations of decentralized spatial algorithms, for use in distributed systems like geosensor networks.

The IN-TORQUE algorithm family relies on two simplifying assumptions that might be relaxed in further work. The assumptions of 2-connected region components and of non-intersecting boundaries are both used to aid routing messages around the region component boundary. In practice, such assumptions will often not hold for real geosensor networks. However, both assumptions might be relaxed by the addition of more sophisticated protocols to ensure a message can traverse (self) intersecting boundaries. Specifically, nodes at a (self) intersection would be required to forward the boundary messages to different destinations dependent on certain boundary and message conditions.

Finally, further work is also required on moving beyond formal models and simulations to testing the IN-TORQUE algorithm family in practical wireless sensor networks and applications. Work on such implementations has begun. Today, the availability and cost of networks with the required level of spatial granularity (i.e., at least thousands of nodes) places severe constraints on current spatial applications of sensor networks. However, it seems certain that technological advances in the near future will enable much larger networks. Indeed preparing for the spatial detail that will be revealed by such networks is one of the central motivations behind the emerging research area of decentralized spatial computing.

Acknowledgments

The authors would like to acknowledge the thorough and constructive comments of three anonymous reviewers, who helped to refine the experimentation and presentation of this paper. We would also like to thank David Bissessar for assisting with some of the data analysis. Matt Duckham's research is funded under an Australian Research Council Future Fellowship (project number FT0990531).

References

[1] S. Nittel, A. Stefanidis, I. Cruz, M. Egenhofer, D. Goldin, A. Howard, A. Labrinidis, S. Madden, A. Voisard, and M. Worboys, "Report from the First Workshop on Geo Sensor Networks," *ACM SIGMOD Record*, vol. 33, no. 1, pp. 141–144, 2004.

[2] Y.-C. Wang, Y.-Y. Hsieh, and Y.-C. Tseng, "Multiresolution spatial and temporal coding in a wireless sensor network for long-term monitoring applications," *IEEE Transactions on Computers*, vol. 58, pp. 827–838, 2009.

[3] N. Lynch, *Distributed Algorithms*. San Mateo, CA: Morgan Kaufmann, 1996.

[4] B. Krishnamachari, D. Estrin, and S. B. Wicker, "The impact of data aggregation in wireless sensor networks," in *Proc. 22nd International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2002, pp. 575–578.

[5] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong, "TAG: A tiny aggregation service for ad-hoc sensor networks," in *Proc. 5th Symposium on Operating System Design and Implementation (OSDI)*, 2002, pp. 131–146.

[6] N. Shrivastava, C. Buragohain, D. Agrawal, and S. Suri, "Medians and beyond: New aggregation techniques for sensor networks," in *Proc. 2nd International Conference on Embedded Networked Sensor Systems (SenSys)*. New York: ACM, 2004, pp. 239–249.

[7] M. B. Greenwald and S. Khanna, "Power-conserving computation of order-statistics over sensor networks," in *Proc 23rd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS)*. New York: ACM, 2004, pp. 275–285.

[8] R. Zheng and R. Barton, "Toward optimal data aggregation in random wireless sensor networks," in *Proc. 26th IEEE International Conference on Computer Communications (INFOCOM)*. Washington, DC: IEEE, 2007, pp. 249–257.

[9] H. Zhang and J. C. Hou, "Maintaining sensing coverage and connectivity in large sensor networks," *Ad Hoc and Sensor Wireless Networks*, vol. 1, pp. 89–124, 2005.

[10] B. Cărbunar, A. Grama, J. Vitek, and O. Cărbunar, "Redundancy and coverage detection in sensor networks," *ACM Transactions on Sensor Networks*, vol. 2, no. 1, pp. 94–128, 2006.

[11] D. Tian and N. D. Georganas, "A coverage-preserving node scheduling scheme for large wireless sensor networks," in *Proc 1st ACM International Workshop on Wireless Sensor Networks and Applications (WSNA)*. New York: ACM, 2002, pp. 32–41.

[12] C. Wen and W. A. Sethares, "Automatic decentralized clustering for wireless sensor networks," *EURASIP Journal on Wireless Communications and Networking*, vol. 5, pp. 686–697, 2005.

[13] J. M. Hellerstein, W. Hong, S. Madden, and K. Stanek, "Beyond average: Towards sophisticated sensing with queries," in *Proc. Information Processing in Sensor Networks (IPSN)*, ser. Lecture Notes in Computer Science, vol. 2634. Berlin: Springer, 2003, pp. 63–79.

[14] R. Gummadi, X. Li, R. Govindan, C. Shahabi, and W. Hong, "Energy-efficient data organization and query processing in sensor networks," *SIGBED Review*, vol. 2, no. 1, pp. 7–12, 2005.

[15] S. Chen, P. B. Gibbons, and S. Nath, "Database-centric programming for wide-area sensor systems," in *Proc. 1st IEEE International Conference Distributed Computing in Sensor Systems (DCOSS)*, ser. Lecture Notes in Computer Science, V. K. Prasanna, S. S. Iyengar, P. G. Spirakis, and M. Welsh, Eds., vol. 3560. Berlin: Springer, 2005, pp. 89–108.

[16] Y. Kotidis, "Processing proximity queries in sensor networks," in *Proc. 3rd Workshop on Data Management for Sensor Networks (DMSN)*. New York: ACM, 2006, pp. 1–6.

[17] M. Demirbas and H. Ferhatosmanoglu, "Peer-to-peer spatial queries in sensor networks," in *Proc. 3rd International Conference on Peer-to-Peer Computing (P2P)*. Washington, DC: IEEE, 2003, pp. 32–39.

[18] D. Goldin, M. Song, A. Kutlu, H. Gao, and H. Dave, "Georouting and delta-gathering: Efficient data propagation techniques for geosensor networks," in *GeoSensor Networks*, A. Stefanidis and S. Nittel, Eds. Boca Raton, FL: CRC Press, 2004, pp. 73–95.

[19] V. Dyo and C. Mascolo, "Adaptive distributed indexing for spatial queries in sensor networks," in *Proc. 16th International Workshop on Database and Expert Systems Applications (DEXA)*, 2005, pp. 1103–1107.

[20] A. Soheili, V. Kalogeraki, and D. Gunopulos, "Spatial queries in sensor networks," in *Proc. 13th ACM International Workshop on Geographic Information Systems (ACMGIS)*. New York: ACM, 2005, pp. 61–70.

[21] K. Park, B. Lee, and R. Elmasri, "Energy efficient spatial query processing in wireless sensor networks," in *Proc. 21st International Conference on Advanced Information Networking and Applications Workshops (IEEE AINAW)*. Washington, DC: IEEE Computer Society, 2007, pp. 719–724.

[22] F. Zhao and L. J. Guibas, *Wireless Sensor Networks—An Information Processing Approach*. San Francisco, CA: Morgan Kaufmann Publishers, 2004.

[23] M. Sharifzadeh and C. Shahabi, "Utilizing Voronoi cells of location data streams for accurate computation of aggregate functions in sensor networks," *Geoinformatica*, vol. 10, no. 1, pp. 9–36, 2005.

[24] P. Skraba, Q. Fang, A. Nguyen, and L. Guibas, "Sweeps over wireless sensor networks," in *5th Int'l Conference on Information Processing in Sensor Networks (IPSN)*, 2006, pp. 143–151.

[25] A. Kröller, S. P. Fekete, D. Pfisterer, and S. Fischer, "Deterministic boundary recognition and topology extraction for large sensor networks," in *Proc. 17th Annual ACM-SIAM Symposium on Discrete Algorithm (SODA)*. New York, NY: ACM, 2006, pp. 1000–1009.

[26] Y. Wang, J. Gao, and J. S. Mitchell, "Boundary recognition in sensor networks by topological methods," in *Proc. 12th Annual international Conference on Mobile Computing and Networking (MobiCom)*. New York, NY: ACM, 2006, pp. 122–133.

[27] J. Lian, L. Chen, K. N. abd Yunhao Liu, and G. B. Agnew, "Gradient boundary detection for time series snapshot construction in sensor networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 18, no. 10, pp. 1462–1475, 2007.

[28] A. Jindal and K. Psounis, "Modeling spatially-correlated sensor network data," in *Proc. 1st IEEE Conference on Sensor and Ad Hoc Communications and Networks (IEEE SECON)*, 2004, pp. 162–171.

[29] G. Jin and S. Nittel, "Towards spatial window queries over continuous phenomena in sensor networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 19, no. 4, pp. 559–571, 2008.

[30] M. Umer, L. Kulik, and E. Tanin, "Kriging for localized spatial interpolation in sensor networks," in *Proc. 20th International Conference on Scientific and Statistical Database Management (SSDBM)*, ser. Lecture Notes in Computer Science, B. Ludäscher and N. Mamoulis, Eds., vol. 5069. Berlin: Springer, 2008, pp. 525–532.

[31] X. Meng, T. Nandagopal, L. Li, and S. Lu, "Contour maps: Monitoring and diagnosis in sensor networks," *Journal of Computer Networks*, vol. 50, no. 15, pp. 2820–2838, 2006.

[32] R. Nowak and U. Mitra, "Boundary estimation in sensor networks: Theory and methods," in *Information Processing in Sensor Networks*, ser. Lecture Notes in Computer Science, vol. 2634. Berlin: Springer, 2003.

[33] R. Sarkar, X. Zhu, J. Gao, L. J. Guibas, and J. S. B. Mitchell, "Iso-contour queries and gradient descent with guaranteed delivery in sensor networks," in *Proc. 27th IEEE Conference on Computer Communications (INFOCOM)*, 2008, pp. 960–967.

[34] I. Solis and K. Obraczka, "Effcient continuous mapping in sensor networks using isolines," in *Proc. 2nd International Conference on Mobile and Ubiquitous Systems (MOBIQUITOUS)*. Washington, DC: IEEE, 2005, pp. 325–332.

[35] S. Gandhi, J. Hershberger, and S. Suri, "Approximate isocontours and spatial summaries for sensor networks," in *Proc. 6th International Conference on Information Processing in Sensor Networks (IPSN)*. New York: ACM, 2007, pp. 400–409.

[36] Y. Xu and W. Lee, "Window query processing in highly dynamic geosensor networks: Issues and solutions," in *GeoSensor Networks*, A. Stefanidis and S. Nittel, Eds. Boca Raton, FL: CRC Press, 2004, pp. 31–52.

[37] M. Duckham, S. Nittel, and M. F. Worboys, "Monitoring dynamic spatial fields using responsive geosensor networks," in *Proc. 13th annual ACM international workshop on Geographic information systems (ACMGIS)*, C. Shahabi and O. Boucelma, Eds. New York: ACM Press, 2005, pp. 51–60.

[38] M. F. Worboys and M. Duckham, "Monitoring qualitative spatiotemporal change for geosensor networks," *International Journal of Geographical Information Science*, vol. 20, no. 10, pp. 1087–1108, 2006.

[39] J. Jiang and M. Worboys, "Detecting basic topological changes in sensor networks by local aggregation," in *Proc. 16th ACM International Conference on Advances in Geographic Information Systems (ACMGIS)*. New York: ACM, 2008, pp. 1–10.

[40] M. J. Sadeq and M. Duckham, "Effect of neighborhood on in-network processing in sensor networks," in *Geographic Information Science*, ser. Lecture Notes in Computer Science, T. Cova, K. Beard, M. Goodchild, and A. U. Frank, Eds. Berlin: Springer, 2008, no. 5266, pp. 133–150.

[41] M. J. Sadeq, "In network detection of topological change of regions with a wireless sensor network," Ph.D. dissertation, University of Melbourne, 2009.

[42] P. Bonnet, J. Gehrke, and P. Seshadri, "Querying the physical world," *IEEE Personal Communications*, vol. 7, no. 5, pp. 10–15, 2000.

[43] M. Egenhofer and R. Fransoza, "Point-set topological spatial relations," *International Journal of Geographical Information Science*, vol. 5, no. 2, pp. 161–174, 1991.

[44] R. Diestel, *Graph Theory*. Berlin: Springer, 2005.

[45] X.-Y. Li, G. Calinescu, P.-J. Wan, and Y. Wang, "Localized Delaunay triangulation with application in ad hoc wireless networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 14, pp. 1035–1047, 2003.

[46] P. Bose, P. Morin, I. Stojmenovic, and J. Urrutia, "Routing with guaranteed delivery in ad hoc wireless networks," in *Proc. 3rd International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications*, 1999, pp. 48–55.

[47] B. Karp and H. T. Kung, "GPSR: Greedy perimeter stateless routing for wireless networks," in *Proc. 6th annual international conference on Mobile computing and networking (ACM/IEEE MobiCom)*. Boston, MA: ACM, 2000, pp. 243–254.

[48] N. Santoro, *Design and Analysis of Distributed Algorithms*. New Jersey: Wiley, 2007.

[49] J. van Leeuwen and R. B. Tan, "An improved upperbound for distributed election in bidirectional rings of processors," *Distributed Computing*, vol. 2, no. 3, pp. 149–160, 1987.

[50] B. Mandelbrot, *Fractals, Form, Chance and Dimension*. San Francisco: Freeman, 1977.

[51] ISO, "ISO/TC 211/WG 2, ISO/CD 19107, geographic information—spatial schema," International Standards Organization, Tech. Rep., 2003.

[52] M. J. Sadeq and M. Duckham, "Decentralized area computation for spatial regions," in *Proc. 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (ACMGIS)*. New York: ACM, 2009, pp. 432–435.

[53] M. F. Worboys and P. Bofakos, "A canonical model for a class of areal spatial objects," in *Proc. Third International Symposium on Advances in Spatial Databases (SSD'93)*. Berlin: Springer, 1993, pp. 36–52.

[54] M. Worboys and M. Duckham, *GIS: A Computing Perspective*, 2nd ed. Boca Raton, FL: CRC Press, 2004. [Online]. Available: http://www.amazon.com/exec/obidos/ISBN=0415283752

[55] G. T. Toussaint, "The relative neighborhood graph of a finite planar set," *Pattern Recognition*, vol. 12, pp. 261–268, 1980.

PLACE PHOTO HERE

Matt Duckham Matt Duckham is an Australian Research Council (ARC) Fellow and senior lecturer in Geographic Information Science at the Department of Geomatics, University of Melbourne. Before moving to Melbourne in 2006, he worked at the National Center for Geographic Information and Analysis (NCGIA) at the University of Maine, USA. His research centers on distributed and decentralized computing with uncertain geographic information, in particular with applications to geosensor networks and ambient spatial intelligence. He is a founding editor of the no-fee, open access journal, the *Journal of Spatial Information Science* (JOSIS) and co author with Mike Worboys of the GIScience text book "GIS: A Computing Perspective".

PLACE PHOTO HERE

Doron Nussbaum Doron Nussbaum received his B.Sc. degree in mathematics and computer science from the University of Tel-Aviv, Israel in 1985, and the M.C.S. and Ph.D. degrees in computer science from Carleton University, Ottawa, Canada in 1988 and 2001, respectively. From 1988 to 1991 he worked for Tydac Technologies as Manager Research and Development. His work at Tydac focused on the development of a geographical information system. From 1991 to 1994, he worked for Theratronics as senior software consultant where he worked on the companys cancer treatment planning software (Theraplan). From 1998-2001 he worked for SHL Systemshouse as a senior technical architect. In 2001 he joined the School of Computer Science at Carleton University as an Assistant Professor. Dr. Nussbaums main research interests are algorithms design, computational geometry, computer gaming, robotics and medical computing.

PLACE PHOTO HERE

Jörg-Rüdiger Sack Dr. Sack received a M.C.S. (”Diplom”) degree from the University of Bonn, Germany, and a Ph.D. from McGill University, Montral, in 1979 and 1984 respectively. His research interests include algorithms, data structures, distributed and parallel computing, geographic information systems and foremost computational geometry.He is co-editor in-chief of the journals Computational Geometry: Theory and Applications, Journal of Spatial Information Science and editor of The Journal of Visualization and Computer Animation. He was awarded an NSERC university-industry Chair in Applied Parallel Computing with a focus on spatial modeling. Most recently, he has been appointed a SUN-HPCVL Chair. He is a founding member of and Carleton’s leading scientist in the High Performance Computing Virtual Laboratory and a member of NSERC’s Committee on Grants and Scholarship where he is Group Chair for Canada’s Computing and Information Sciences. He is also serving on the joint committee at the DFG (German Research Council) for the Excellence Initiative and is a member of the G8 Research Councils Initiative on Multilateral Research Funding: Exascale Computing .

PLACE PHOTO HERE

Nicola Santoro Nicola Santoro is Professor of Computer Science at Carleton University. He has been involved in distributed computing from the beginning of the field, contributing extensively on the algorithmic aspects. He is a founder of the main theoretical conferences in the field (PODC, DISC, SIROCCO). He has authored the book ”Design and Analysis of Distributed Algorithms” (Wiley 2007) and he is recipient of the 2010 ”Prize for Innovation in Distributed Computing”. His current research is on distributed algorithms for mobile agents, autonomous mobile robots, and mobile sensor systems.