

Topological Data Processing for Distributed Sensor Networks with Morse-Smale Decomposition

Xianjin Zhu Rik Sarkar Jie Gao

Department of Computer Science, Stony Brook University. {xjzhu, rik, jgao}@cs.sunysb.edu

Abstract—We are interested in topological analysis and processing of the large-scale distributed data generated by sensor networks. Naturally a large-scale sensor network is deployed in a geometric region with possibly holes and complex shape, and is used to sample some smooth physical signal field. We are interested in both the topology of the discrete sensor field in terms of the sensing holes (voids without sufficient sensors deployed), as well as the topology of the signal field in terms of its critical points (local maxima, minima and saddles). Towards this end, we develop distributed algorithms to construct the Morse-Smale decomposition. The sensor field is decomposed into *simply-connected* pieces, inside each of which the sensor signal is *homogeneous*, i.e., the data flows uniformly from a local maximum to a local minimum. The Morse-Smale decomposition can be efficiently constructed in the network locally, after which applications such as iso-contour queries, data-guided navigation and routing, data aggregation, and topologically faithful signal reconstructions benefit tremendously from it.

I. INTRODUCTION

In this paper we are interested in the topological features of spatially distributed sensor data. In many application settings such as environmental monitoring, the sensor readings can be regarded as a dense sampling of an underlying physical signal field that often exhibits strong spatial and temporal correlations. Such spatial characteristics are important for many applications of sensor networks, as they correspond to physically significant phenomena. For example, peaks indicate heat sources in a heap map or traffic jams in car density map. Users of sensor networks are often interested in data-related queries such as

- *Iso-contour query*: from a query node q , find the iso-contours at value x , or count/report iso-contour components at a given value/range. This can be used to discover for example high pollution areas for rescue workers, or group targets for police officers.
- *Data-guided navigation and routing*: find a path from a source node s to a destination node t with all values on the path within a user-specified range. This can be used for navigation of packets in the network (e.g., avoiding sensor nodes with low energy level), or navigation of users/vehicles in the physical environment (e.g., avoiding traffic jam).

The spatial features such as local maxima, minima or saddles and their relationship capture the topological complexities of the signal field. Abstractions of these spatial features will allow aggressive data compression and reconstruction while still preserving the important topological characteristics, which is useful for efficient data delivery and storage.

Motivated by the significance of the topological spatial features in the distributed sensor data, we need to come up with algorithms for extracting and representing these features. One major challenge we face immediately is the identification of critical points, especially the saddles. In a continuous signal field, a critical point p is a point with all partial derivatives vanishing at p — but with a discrete sampling almost surely there is no sensor with exactly the same value as its neighbors. We can identify the local maxima/minima as the nodes with all neighboring values no greater (smaller) than themselves. However, it is not easy to robustly identify saddle points with increasing and decreasing neighbor values in an alternating fashion, specially when we do not have accurate sensor locations and angular orderings of neighbors (self-localization in a large sensor network is still a research challenge in the sensor network community).

In our previous work [12] we developed a distributed algorithm to extract the contour tree, which is a tree on all the critical points of the signal field and captures how the connected components of the iso-contours merge/split as we increase/decrease the isovalue. With the notion of contours, a node is identified as a saddle when contours start to split or merge the first time. We also developed sweep-based algorithms to advance the contours and identify the saddles on the way, as well as gradient based greedy routing algorithms for iso-contour queries and data-guided navigation.

There are a number of limitations to the contour tree approach [12]. First the contour tree construction algorithm assumes a sensor network deployed in a region without holes. Otherwise a contour might be broken into multiple disconnected pieces that advance by themselves. It is then much harder to synchronize these different pieces of frontiers to identify the saddle point, and the competition resolution mechanism used in [12] might incur a high communication cost by delivering messages between these pieces back and forth. Second, the current contour tree algorithm assumes a static sensor data field and does not easily adapt to a time-varying signal field. The problem of designing robust algorithms for a time-varying signal field, specially in the distributed network setting, still remains open.

In this paper we propose a perpendicular approach to handle spatial signals. We deal with the topological structures of the signal field (in terms of critical points) and the topological structures of the sensor field (in terms of holes) simultaneously. We apply Morse theory [11] in sensor network setting and propose a communication-efficient distributed algorithm to

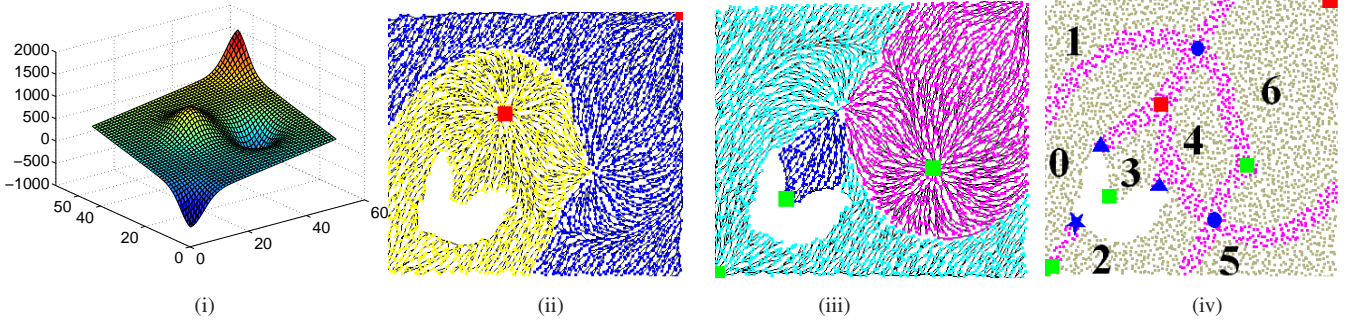


Fig. 1. (i) Original signal field. The network has a hole. (ii) Stable manifolds. (iii) Unstable manifolds. (iv) Morse-smale decomposition with each cell labeled. Red square: *max*; green square: *min*; blue disk: *regular saddle*; blue star: *max-saddle*; blue triangle: *min-saddle*.

decompose a sensor network to cells. Each cell is simply connected (i.e., has no holes) and homogeneous (i.e., the data flows uniformly from a local maximum to a local minimum). The cell adjacency information is captured and represented by the Morse-Smale complex, which is a compact structure with size proportional to the number of critical points in the signal field and the number of holes in the network. One can thus afford to disseminate the Morse-Smale complex to all sensor nodes in the network as a high-level summary of the signal topology. The homogeneous flow inside each cell gives a natural coordinate system with one set of coordinates along the greatest descent vector and the other set of coordinates along the isolines, both of which interweave nicely as a Cartesian coordinate system in the cell, and are smoothly glued along the boundary with adjacent cells. Thus the coordinate system supports local and easy navigation or routing operations both inside and across the cells, that can be exploited by the iso-contours queries and the data-guided navigation. Together with the compact Morse-Smale complex available at each node, we immediately have a 2-level routing structure, akin to the virtual coordinate system built in the GLIDER algorithm for efficient point-to-point routing [3], such that a global routing decision (e.g., a value-restricted routing request) can first consult with the high-level structure to identify the cells to visit, with the actual routing implemented with this global guidance by local greedy routing scheme inside each cell. This achieves a nice balance in supporting functions requiring global information through local operations under energy conservation requirement.

In the remaining part, we briefly review Morse theory and then elaborate the contribution in this paper that involves non-trivial distributed algorithm design for Morse-Smale decomposition and its applications in iso-contour queries, data-guided navigation and routing, data aggregation, and topologically faithful signal reconstructions.

Morse-Smale Decomposition theory. Morse theory [14], [15] deals with the relation between the topology of a smooth manifold and the critical points of a smooth real-valued function f defined on the manifold. A point p is a critical point if the tangent vector at p is zero. Following the gradient vectors, a *stable manifold* of a critical point a is defined as the union of a and all points flowing into a . Similarly, an *unstable manifold* of

a critical point a is the union of a and all points flowing out of a . A Morse function f is called Morse-Smale if the stable and unstable manifolds intersect only transversally. In this case, the Morse-Smale decomposition is the intersection of the stable and unstable manifolds. Each cell in the decomposition is a quadrangle with a local maximum, two saddles and a local minimum. This means all the gradient vectors in a cell are *uniform* – they all originate from the same maximum and flow into the same minimum. The Morse-Smale complex takes the dual of the decomposition of Morse-Smale cells and captures the topology of \mathcal{M} through the study of the gradient of f .

When we apply Morse theory to a sensor field with holes, the good properties mentioned above do not directly carry over. The network holes can disrupt the Morse-Smale decomposition (for f defined in \mathbb{R}^2) in the sense that a cell may contain one or multiple holes in its interior and is no longer simply connected — thus causing problems with the Cartesian coordinate system as greedy routing can get stuck at the hole boundary and no longer deliver the message successfully. More critical points might be introduced by the holes as flows may end or start from hole boundaries.

In a companion paper [8], we developed the theory for Morse-Smale decomposition in a 2D region with boundaries and restored the good properties of such decomposition. Of special interest to the sensor network application is that we establish the connection of the saddle points with the ‘cut locus’ of the sensor data flow. The notion of cut locus is originally defined as the collection of points with two or more geodesic paths to the same root with different homotopy types (i.e., getting around the holes in different ways) [1]. In an earlier work [16] we used the shortest path map from one point in the sensor domain to discover the topology of the sensor field. In this paper, we utilize the natural gradient flow of f , to capture both the topological features of the underlying domain, as well as the topological structures of the signal field. Essentially the flows along the greatest gradient vectors have limit endpoints at the critical points (local maxima, minima or saddles). Thus we identify a pair of *cut nodes* as two neighboring sensors with two different flows, either arriving at two different maxima/minima, or at the same maximum/minimum with different homotopy types (i.e., bypassing network holes in different ways). The cut pairs leading to different maxima/minima represent the boundaries

of the stable manifolds of these different maxima/minima. The cut pairs leading to the same maximum/minimum but through flows of different homotopy types will further cut the holes open to make each cell homogeneous and simply connected. See Figure 1 for an example. The interpretation of the Morse-Smale decomposition by cut locus of the flow lines allows us to use the local algorithm in [16] to identify saddle points, which is more robust and saves communication cost.

The algorithm for constructing the Morse-Smale decomposition and its dual for a sensor network does not require any information of where the holes are and how many of them. In fact, hole or boundary detection of a location-free sensor network is by itself an interesting and challenging research problem [6], [10], [16]. The Morse-Smale decomposition and its dual complex provide a new approach to derive the homology of the sensor network, by using the natural signal field on the sensors, different from previous approaches with simplicial homology or witness complexes [7], [9].

Contribution in this paper. Our companion paper [8] mainly laid out the rigorous definition for Morse-Smale decomposition for a 2D continuous region with holes and proved the equivalence to the cut locus of the data field, thus providing the theoretical foundation. This paper discusses the detailed implementation in a discrete network setting. The same as many other work on geometric algorithms in sensor networks, the adaptation of notions and ideas from a continuous domain to the discrete network setting often requires careful thinking and non-trivial algorithm design techniques, to come up with communication-efficient algorithms that are robust to discreteness, noises and dynamics in the data. In the literature, an algorithm for computing the Morse-Smale complex of a function defined on a piecewise linear surface has been developed by Edelsbrunner, Harer and Zomorodian [2], by mimicking the smooth setting with the idea of simulation of differentiability. A combinatorial Morse theory has also been developed with discrete gradient vector fields [4], [5]. These work, unfortunately, can not be directly applied to sensor network setting as we do not have any piecewise linear mesh (as in [2]) nor a well-defined cell complex (as in [4], [5]) to work with. It is in fact non-trivial to construct or maintain any time of well-structured mesh or cell complex when sensors may fail and communication links may go up and down constantly. In addition, we ask for an distributed algorithm that does not require global knowledge or coordination.

In this paper we implement the Morse-Smale decomposition with the help of the cut locus of the flow, suggested by the theoretical results in [8], and carry out all the details in a networking environment with possible link/node failures and data dynamics. We also present simulations results on the communication cost of the construction.

Another major contribution in this paper is the application of the Morse-Smale decomposition for sensor network problems:

- *Iso-contour queries and data-guided routing.* As explained earlier, the homogeneous flow inside each cell makes these routing primitives very easy with only local

greedy decisions.

- *Sweeps for data collection and aggregation.* Sweeps are used as a basic data collection and aggregation scheme in a sensor network [13]. Data is collected with the sweep frontier that progresses according to some potential function. A straight-forward choice of the potential function is the signal field itself. However, the presence of holes or saddles may tear apart the sweep frontier and even worse, have one piece waiting for the other pieces before it can proceed. With the Morse-Smale decomposition we can perform a sweep inside each cell that is simply connected, which guarantees the connectivity of the sweep frontier, thus reducing a lot of time on sweep frontier coordination and synchronization.
- *Topologically faithful compression and reconstruction.* If one records the values and positions of the nodes in the Morse-Smale decomposition boundary, it is easy to perform linear interpolation to reconstruct a signal field with precisely the same topological features. The reconstruction is not meant to be geometrically close but nevertheless can provide all the information that concerns the signal and the network topology.

We describe in details how to make use of the Morse-Smale decomposition in these applications and provide performance improvement results in simulations.

In this paper we mainly assumes a static signal field to discuss the algorithm and applications. Nevertheless, the Morse-Smale decomposition adapts to dynamic signal field better than the contour tree, as the flows and gradient vectors can be maintained locally. We briefly touch upon these issues in Section II. A thorough evaluation in a dynamic signal field remains as future work.

II. MORSE-SMALE DECOMPOSITION

We briefly go over the steps to get a Morse-Smale decomposition and then elaborate on each part. We first define gradient vectors for a discrete sensor data field. That is, each node computes an ascent and descent vector to one of the neighbors leading to the upstream node and the downstream node along the flow. Local maxima/minima can be easily identified locally as they do not have an ascent/descent vector. Now, to get the Morse-Smale decomposition, we make use of the theory in [8] that the cut locus of the flow (defined as the pairs of neighboring nodes whose flow go to different critical points or go to the same critical point bypassing holes in different ways) is equivalent to the boundary of the Morse-Smale decomposition (Theorem 2.18 in [8]). The algorithm for detecting cut locus borrows from [16] but the idea in [16] can only detect part of the cut locus in our setting. We will extend the partial cut locus to the full cut locus by taking the nodes that flow into them. In the last step we get the Morse-Smale decomposition with each cell identified and given an ID. The cell adjacency information is collected and disseminated to all the nodes in the network to facilitate topological data analysis.

Compute ascent and descent vectors. Nodes with locally maximum and minimum values can be easily identified by

comparing with neighbors' values. The set of neighbors of a node p is denoted $N(p)$.

Definition 2.1. MAX and MIN. A node p is identified as MAX, if $V(p) > V(q)$ for each node $q \in N(p)$, where $V(p)$ is the value of node p and $N(p)$ is the 1-hop neighborhood of p . p is identified as MIN if $V(p) < V(q)$ for each node $q \in N(p)$.

Here we assume no nodes have equal values and will discuss the plateau case later.

At the same time, each regular node can automatically discover an upward flow towards a unique MAX and a downward flow towards a unique MIN. The definition of flow is based on the definition of gradient. In the continuous case, gradient basically defines the steepest ascent and descent directions in a signal field. In the discrete case, we define ascent and descent vectors accordingly.

Definition 2.2. Ascent and descent vectors. For a sensor node p , if there is a neighbor q , $V(q) > V(p)$ and $V(q) > V(q')$ for any $q' \in N(p)$, p maintains a pointer to q , which is called an ascent vector. Similarly, if $V(q) < V(p)$ and $V(q) < V(q')$ for any $q' \in N(p)$, the pointer is called a descent vector.

Each sensor node p can compute its ascent and descent vectors by locally checking the value $V(q)$ of each neighbor q . If location information is available, we can refine these vectors by normalizing the difference of values of p, q with the distance between p and q , then taking the neighbors with the greatest ascent and descent gradients. It is easy to see that every node has both ascent and descent vectors except local maximum and minimum nodes. Local maximum nodes only have descent vector and local minimum nodes only have ascent vector.

With the ascent and descent vectors locally identified, upward and downward flow are implied immediately. Basically, upward flow through any node climbs up along the ascent vector and eventually flows into a MAX. The downward flow goes down along the descent vectors to a MIN. Each node p can follow the orbit in form of successive ascent/descent vectors and determine the unique $\max(p)/\min(p)$ it flows to.

Identify cut locus. As explained earlier, saddles are especially hard to detect due to the discrete nature of a sensor field. Complex geometric features like holes of the network will pose additional challenges to this task. In the original Morse theory, the saddles and the flows into these saddles represent the boundary of the Morse-Smale decomposition. Our theoretical results in [8] says that the boundary is in fact equivalent to the cut locus of the flow, thus allowing a rather robust method to identify the saddles and the Morse-Smale decomposition boundary.

In the discrete setting, we define the cut locus as the collection of pairs of nodes, such that each pair has flows of different homotopy types, or different endpoints (MAX or MIN).

Definition 2.3. Cut pair and cut locus. A neighboring pair of

nodes p and q is a cut pair if (i) $\max(p) \neq \max(q)$ or $\min(p) \neq \min(q)$; or (ii) p and q flow to the same critical point but their flows have different homotopy types. The union of cut pairs forms the cut locus.

Some cut pairs can be identified locally. A node p can easily check if there is a neighbor q such that $\max(p) \neq \max(q)$ or $\min(p) \neq \min(q)$. To detect a pair of nodes whose flows reach the same \max/\min but have different homotopy types, we adopt the local cut detection algorithm proposed in [16]. The method in [16] detects cut nodes for the geodesic distance function with respect to a single root. Here, we wish to detect cuts with respect to upward/downward flows. We describe the algorithm using the upward flow. The same idea works for downward flow.

A pair of neighboring nodes u and v check if they have flows of different homotopy type, i.e., enclosing one or more holes in between, based on two parameters δ_1 and δ_2 , which specify the minimum size of a hole. δ_1 and δ_2 are typically chosen as some constants. u, v detect themselves as a cut pair if the following conditions are met: 1) their lowest common ancestor along the flows is at least hop δ_1 away; 2) the distance between the flows of u and v is at least δ_2 . A node can cache δ_1 nodes along its flow to the max or can gather such information on-demand by following ascent vectors for δ_1 hops. Thus, u and v can exchange their upward flow information and determine if the first condition is satisfied. To efficiently measure distance between two flows, we only check the distance between a pair of nodes a and b $\delta_2/2$ away from u and v along their flows respectively. If a and b are at least δ_2 hops apart, we say the second condition is satisfied. By this algorithm, we detect precisely the cut pairs close to the hole boundary, thus in the neighborhood of a saddle. See Figure 2 (i). The rest of the cut locus can be detected by finding the set of nodes flowing into these saddles. For example, see the pair in Figure 2 (ii). It is on the cut locus but may not be able to detect themselves locally, as their flows move forward side by side until they hit the hole boundary where they depart. If the pair is of distance at least $\max(\delta_1, \delta_2)$ away from the hole boundary, this pair can not be discovered by our local cut locus detection algorithm. But the flows from this pair (u', v') will go through the pair (u, v) and will be detected then. This idea is refined as below and saddles are identified and classified.

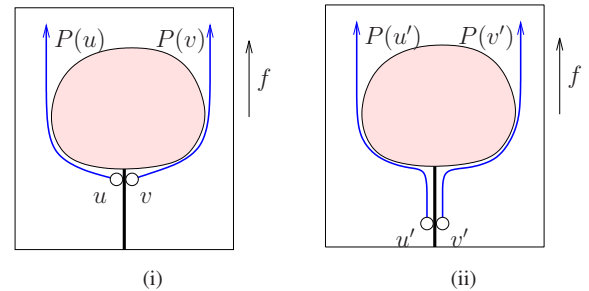


Fig. 2. Suppose the function f is defined as the function $f(x, y) = y$ with x -coordinate as the horizontal axis and the y -coordinate as the vertical axis on the paper. Intuitively the function f flows bottom to top. The cut pair (u, v) in (i) will be detected by the local cut locus detection algorithm. The cut pair (u', v') will be detected as they flow into (u, v) .

Identify saddles and refine cut locus. There are three types of saddles, i.e. regular saddle, max-saddle, min-saddle. All of them can be detected locally based on cut locus.

Definition 2.4. Regular saddle, max-saddle and min-saddle. Regular saddles are saddles of a signal field with no network holes and identified by a pair of neighboring nodes flowing into different critical points. A saddle is a max-saddle if it is generated by a hole and identified by a pair of cut nodes with different homotopy upward flows to same or different max. Similarly, a saddle is a min-saddle if it is generated by a hole and identified by a pair of cut nodes with different homotopy downward flows to the same or different min.

When detecting cut locus, cut pairs also record their identifiers. If a pair of nodes p and q are detected as cut because $\max(p) \neq \max(q)$ or their upward flows have different homotopy types, we say p and q are identified by upward flows, and $\max(p)$ together with $\max(q)$ (that may or may not be the same) are their identifiers. Otherwise, the cut pair is identified by downward flows, and $\min(p)$ together with $\min(q)$ are their identifiers. For a connected component of cut nodes identified by upward flows with the same pair of identifiers, we detect the *max-saddle* by selecting the node with the highest value in the connected component. Similarly, we detect *min-saddle* by selecting the node with the lowest value in the connected component. If a node is identified both *max-saddle* and *min-saddle*, then it becomes a regular saddle. There are sometimes multiple nearby nodes identified as regular saddles. We group them together and select one as a representative (e.g. select the node with the highest ID).

Recall that when detecting cut locus, we may only detect cut nodes around hole boundaries, which are within close neighborhood of *max-saddle* and *min-saddle*. To restore the part of cut locus that have not been detected directly by the algorithm above, we start from the *max-saddle* and *min-saddle*, and extend the cut locus by including the nodes flowing into and flowing out from those saddles. More specifically, a *max-saddle* sends an invitation message along the downward flow pointer. Every node p upon receiving the message checks if the *max-saddle* is on the path of its upward flow. If this is the case, p marks itself as cut node. To make the cut locus connected and fully partition the nodes in different cells, p also include its one-hop neighbors as cut nodes. Cut locus is also extended from *min-saddle* in the similar way. Eventually, the refined cuts stop at either network boundaries or other critical nodes. A cut component is formed by cut nodes between two critical nodes (or boundaries) and assigned a unique cut component ID. Critical nodes can belong to multiple cut components.

Morse-Smale decomposition. The cut locus essentially becomes the boundaries of Morse-Smale cells. The nodes in the same cell will be recognized and given a unique cell ID. To do that, each max will initiate a flood message to a k -hop neighborhood of the max, with k as a small constant. With the nodes in cut locus removed, the connected components

in the k -hop neighborhood will identify themselves as in different cells. They will propagate the message further with their cell IDs to all the other nodes in the same connected component to identify all the nodes in the same cell. For a pair of cut nodes that stay on the cell boundaries, we ensure the left cut node belongs to the left cell and the right cut node goes to the right cell. To do that, we assign the cut nodes to a unique cell according to what cell the other nodes on their upward/downward flow is assigned to. Specifically, a cut node reaching a max-saddle searches along its upward flow and joins the cell of nodes along the flow. Same for the cut locus for a min-saddle. It is possible that maximum and minimum nodes belong to multiple cells. They join every cell that their neighbor belongs to.

Construct the dual complex. Finally, to build a compact representation to capture the topology of both the signal field and the network, we construct the dual Morse-Smale complex of the decomposition and store such compact structure at each sensor node to facilitate upper-level applications. Appropriate information and application-dependent labels can be stored for each simplex of the dual to facilitate information retrieval. For example, for the routing applications, we include with each edge of the complex the max and minimum value of the nodes on the boundary of the two cells. For the signal compression and reconstruction application, we will include the positions/values of a chain of nodes on the boundary of two cells, from the maximum to the minimum.

Only the cut nodes are involved in constructing the morse-smale complex. Cut nodes can detect adjacent cells by checking the cells of neighboring non-cut nodes. A cut node can be adjacent to multiple cells. Each endpoint of cut components (critical nodes or boundary nodes) constructs a morse-smale complex based on its local information and propagates the information together with its ID through cut locus to other endpoints. Faces of morse-smale complex are annotated with the original detector ID. When an endpoint p receives information from another endpoint q , it combines with its own information and removes duplicate faces if their detectors belong to the same cut component. The complete morse-smale complex is constructed after gathering information from all cut components. This compact structure can possibly be propagated and stored at each sensor node depending on application requirements.

Noise handling and dynamic signal field. When a sensor data field is noisy, it is possible that multiple nearby nodes become local max or local min. For example, if a hole boundary coincide with an isocontour around a peak of a signal field, all the nodes on the hole boundary become local maximum. Or when the signal field itself has a noisy plateau region there can also be multiple nearby nodes that all identify themselves as local max. To handle these noises, we parameterize our algorithm with a tolerance factor δ , cluster the δ -hop neighborhood of node p as its direct neighbors, and apply the algorithm on this hyper one-hop neighborhood. All definitions are defined as before and computed accordingly. A

node becomes local maximum/local minimum only if its value is the highest/lowest in its δ -hop neighborhood. Similarly, the ascent/descent vectors are also chosen from this δ -hop neighborhood, and so are the flows and cut locus.

We also remark that the Morse-Smale decomposition, unlike contour tree, has a better handling of dynamic data. This is because the gradient ascent/descent vectors are defined and detected locally, and thus can be updated locally as well when the data varies over time. When a node changes its value, it propagates the new value to its neighbors and adjust its ascent/descent vectors. When a new max/min appears, it obtains a new ID and propagates along the downward flows. When an existing max/min hops to a neighboring node, the upward/downward flow carries the current max/min ID, and eventually gives the ID to the new max/min when the upward/downward flow stops. When new cells or the cell adjacency information changes, the Morse-Smale complex will be updated. We currently propose to periodically update the Morse-Smale complex stored at the nodes. Further investigation on a dynamic data field by exploiting the full potential of the Morse-Smale decomposition will remain as future work.

Features of Morse-Smale decomposition. We summarize the nice features of morse-smale decomposition as follows. In the next section these features are used to develop application-dependent data structures for topological analysis of distributed sensor data.

- The cells bounded by cut locus are equivalent to morse-smale decomposition.
- Each cell is simply connected. Inside each cell, the signal field is homogeneous, i.e., flows are uniformly from the unique maximum to the unique minimum.
- The dual morse-smale complex captures the topology of both the signal field and the network. It is homotopy equivalent to the network.

III. APPLICATIONS

In this section we describe the benefits of having a Morse-Smale decomposition and dual complex in different applications in a sensor network. In particular, describe how the decomposition facilitates data-centric routing, aggregation by sweeps and signal reconstruction by decomposing the network into simple monotone pieces. Throughout the discussion, we assume that the dual complex has been computed and is known at every node.

A. Data centric routing

We are interested in two types of data dependent queries :

- **Value restricted routing.** Find a path from a source node s to a destination node t with all values on the path within a user-specified range.
- **Iso-contour query.** From a query node q , find the iso-contours at value v , or count or report iso-contour components at the given value or range. This involves local determination of contour components and routing to each of these components.

The following concepts will be useful in both the discussions.

Definition 3.1 (Dual path). Given cells c_s and c_t , a dual path between them is a simple sequence $c_s = p_1, p_2, \dots, p_n = c_t$ such that $\forall i, (p_i, p_{i+1})$ is an edge in the dual, that is the corresponding cells are neighbors.

This implies a top level path from source cell to destination cell. In the implementation, we avoid neighboring cell pairs whose intersection is a single point, and take only those neighbors that share an edge. This is done to avoid overloading the single shared point with routing traffic between the two cells. For the same reason, we select a random dual path instead of a deterministic one.

Definition 3.2. Routing primitives

- **Nodes on a contour v .** A node p is said to be on a contour v if the value at p is v , or if the value of p is greater than v , and p has some neighbor whose value is less than v . This definition establishes the contours in terms of nodes in the network and will be used in routing and detecting iso-contours.
- **Routing to an iso-contour v .** Given a node p in a cell c_p , routing to a contour within the cell consists of routing to a random higher neighbor of p iteratively, until a node on the contour v is reached.
- **Routing to a neighboring cell.** This is executed in two steps. First, from the union of ranges of edges shared with the neighboring cell, select a random contour value, and route to a node on that contour. Next, follow nodes on the contour to reach the destination neighbor cell. This is possible, since by the properties of Morse-Smale decomposition, each contour level intersects a cell in a single connected component. The correct direction to follow along the contour can be determined by storing a small amount of orientation information along each contour, equivalent to local coordinates.
- **Routing to a node q in the same cell.** This is achieved by first reaching an iso-contour at the value of q , and then following the iso-contour to reach q .

With these definitions in place, it is now possible to route from any node to an arbitrary cell in the network, by first computing a random dual path, and then traversing the dual path by means of the low-level routing primitives described above. It is also possible to route to a specific node in the network, by using the fourth primitive as a final step.

We apply randomization in dual path selection, and in routing within a cell. This produces different paths between every source-destination pair on each query (see fig. 3 (i)). This randomization helps distribute the routing load in the network.

1) *Value restricted routing:* This is done simply by suitable restrictions on the routing primitives described above. Given a routing request, with values restricted to $[a, b]$ we perform all the routing steps relative to this range. In particular:

- 1) In finding a dual path, only consider edges whose range has a non-empty intersection with $[a, b]$.

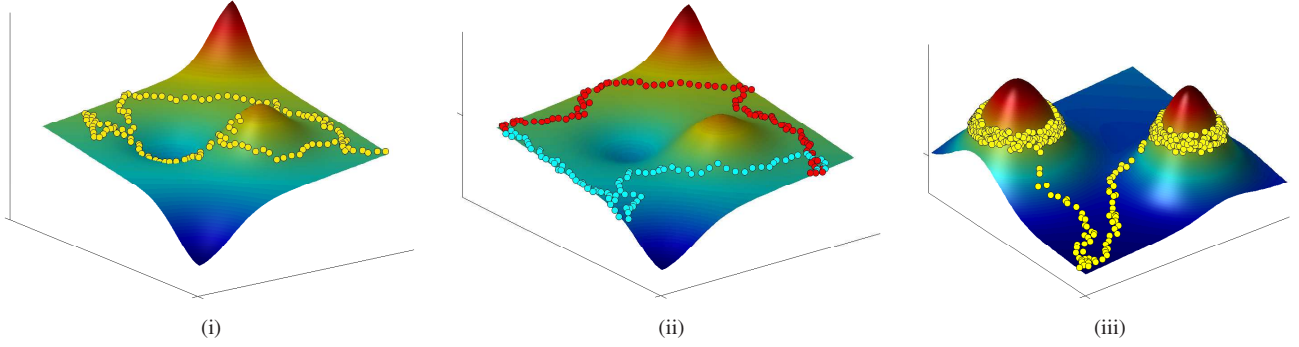


Fig. 3. (i) Two very different routes are utilized on two different routing requests. This result of randomization helps in load balancing. (ii) Depending on the range restriction applied, the algorithm constructs different paths. The red path is in response to a request for a path in a *high* range, the blue path on request for a *low* path. (iii) Iso-contours and routes to the different iso-contours found by the algorithm.

2) In routing to a neighboring cell, select randomly a valid contour level, as before, but restricted to the range $[a, b]$.

It is easy to see that these two adjustments suffice to perform value restricted routing. Figure 3 (ii) shows a case where different range restrictions result in different paths.

2) *Iso-contour queries*: Iso-contour queries can be executed in two steps:

Find and group iso-contour cells. First, identify the cells that contain some portion of the iso-contour in question, this can be done simply observing the maximum and minimum values in each cell. Now, several such cells may contain parts of a single connected contour component. We can identify the disjoint components using the following principle : *two of the already identified cells are neighbors and they share an edge whose range contains the queried level, iff the corresponding iso-contours belong to the same connected component*. This fact can be used to identify the connected components by a simple DFS in time linear in the size of the dual graph.

Route to each connected component. Once the connected components of iso-contours have been identified, we can select one cell of each component and route to it and to the contour level by the methods above. Figure 3 (iii) shows an example.

In this subsection we have presented examples on simple networks without *holes* for ease of understanding and graphic representation. The routing mechanisms however, are independent of such issues, and work equally well in presence of holes.

B. Data aggregation by sweeps

The idea of using a sweep over a sensor network to perform data aggregation was proposed in [13], where this method was shown to be more efficient than standard aggregation tree based methods.

The intuition behind performing a sweep is to schedule the transmissions in a way that reduces collisions and energy usage. In the initial phase of the algorithm, an ordering is established on the nodes, which is equivalent to defining a real valued function f over the domain. In the second phase, the sweep is executed. Each node that is a local maximum of f , invites its lower neighbors to join the sweep. A node waits for invitations from all its higher neighbors, and once all such invitations have been received, it starts sending invitations to its lower neighbors. Once a node has sent all required

invitations, its part in the sweep is done and it can retire and turn itself off. Therefore, the time that a node must stay *active* is from the reception of its first invitation to the transmission of the final invitation. Minimizing this up-time can greatly reduce power consumption and improve network life time.

The function f which determines the sweep can be an arbitrary one, with the understanding that the sweep will end at the minima of this function. It can simply be one of the geographic coordinates, producing a uniform slope (Fig. 4(i)) or a harmonic potential (see [13] for details) or simply an existing signal field (Fig. 4(ii)). The advantage of this last method is that it requires no expensive pre-computation of harmonic potential, nor does it require node locations. We assume here that the sweep is performed with respect to such a given arbitrary function, that may be natural or artificial and it suffices to terminate the sweep at the minima.

A disadvantage of the sweep method is that it pauses at the saddles. The sweep frontier, consisting of connected components of active nodes corresponds to iso-contour components of the function f . At a saddle where different components of the sweep merge into a single iso-contour, it is likely that one sweep component will arrive before the other, and active nodes will have to wait for the other component to arrive. Figure 4(i) shows an example where at a certain stage of the sweep, the sweep frontier near a hole in the network has to wait in active state for the rest of the sweep to arrive. Similar situations can occur while sweeping by natural signal as well (Fig. 4).

To overcome this issue, we sweep the network with the help of the Morse-Smale decomposition. That is, we sweep each cell of the network independent of the others. Since Morse-Smale cells do not contain saddle points, idle pauses for synchronization of sweep are not necessary. This saves much of the active time of nodes and speeds up the aggregation. Figure 4(iii) shows the status in this case after an equal interval of time. The sweep frontiers in this case are all actively moving in their segments. Further, a larger number of nodes have been swept, implying a faster process in general. The sweep can be seen to proceed in several independent components in segments around the network. This parallelism greatly decreases the overall aggregation time, and reduces MAC layer collisions and interference. Quantitative ideas of these benefits will be described in the simulations section.

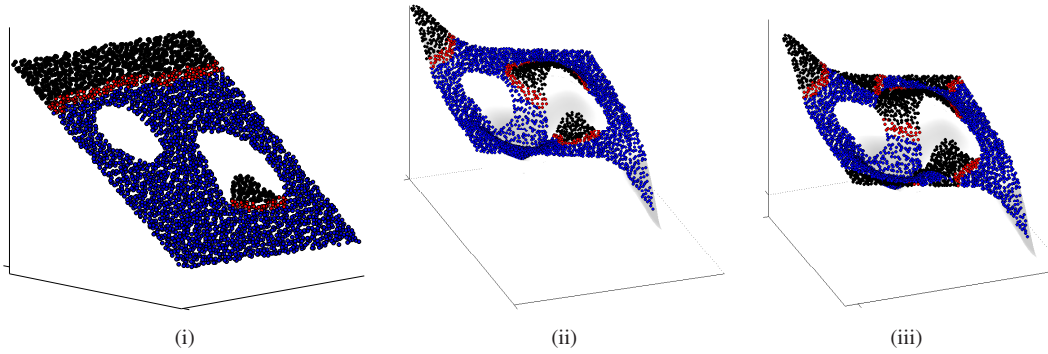


Fig. 4. Sweep in progress under different environments. *Black: swept nodes; blue: nodes not swept yet; red: active nodes.* (i) Shows a sweep by a geographic coordinate, where one sweep frontier has to wait at a saddle for another frontier to arrive. (ii) shows a similar case, but in sweep according to a signal value. (iii) Shows a sweep on the Morse-Smale decomposition of the same signal. Many more nodes have been swept in this case. All states are after an equal interval of time on a global clock.

C. Topology faithful compression and signal reconstruction

We briefly discuss the signal compression value of summary information obtained from Morse-Smale decomposition. Note that the properly labeled dual complex itself encodes most of the topological properties of the signal. It represents all saddles, maxima and minima. The *contour tree* and related information for example, can be extracted from this data. Making it possible to extract high level routes, as described above. The complex in addition also encodes topological properties of the network itself, in particular presence of holes, and their relation to the critical points.

In some cases however, it may be desirable to obtain slightly more geometric information about the signal. While the dual declares the presence of Morse-Smale cells that are guaranteed to be simple and homogenous, it says nothing about the shape of these cells. The idea in this section is that in a network with locations, some geometric aspects of the decomposition can be obtained at a low cost. This can be done simply by tracing the cut locus, that is, the boundaries of cells and the maxima and minima. Given a path along the boundary, whose node locations are known, we now have the network decomposed into cells of known boundaries, that are guaranteed to be Morse-Smale cells. Therefore any interpolation that satisfies the properties of such a cell will represent a topology faithful reconstruction.

Figure 5 shows the reconstruction information available from such a boundary extraction. For each cell, we simply represent each of the connected components of its boundary by a single path. A network of about 4000 nodes, in this case is represented by chains of cut locus of only 300 node locations and values. The interesting feature of the reconstruction information is that all the *important* points of the original signal have been represented.

IV. SIMULATIONS

We present here simulation results demonstrating performance of the routing and aggregation applications when executed on top of the Morse-Smale decomposition.

Computation of Morse-Smale Decomposition and Iso-contour Queries. We compare the performance of iso-contour queries with the contour tree approach [12]. The signal function and networks used in this section are identical to those

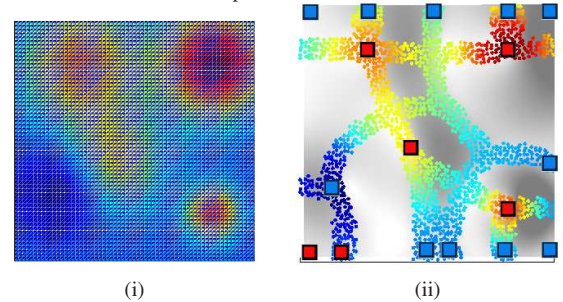


Fig. 5. Reconstruction. (i) original signal. (ii) Reconstruction information from paths along the cut locus, encompassing critical points of the signal. Maxima and minima are shown as red and blue points respectively.

in [12]. These networks do not have holes, since the contour tree construction does not handle holes well. Our algorithms however would also operate successfully on networks with non-trivial topologies.

We first sampled the function in Figure 6(i) with networks of varying number of nodes. The result in Figure 6(ii) shows that the communication cost for the Morse-Smale decomposition is linear in network size and therefore scalable.

Next, in a network of 1600 nodes, we ran iso-contour queries, from nodes chosen uniformly randomly from the network, and querying for a random signal value between the highest and lowest in the network. As in [12], we compare the query cost with that of a centrally computed minimum spanning tree, and plot the CDF of this ratio. Figure 6(iv) shows the CDF of the loads on network nodes. These graphs are found to be comparable or better than those of the contour tree method.

Aggregation by sweep. We described in the previous section why the decomposition can improve the efficiency of data aggregation by sweep. In this section, we describe simulation results that confirm our claim. Sweeps were simulated on the networks shown in fig. 7. An elementary medium access model

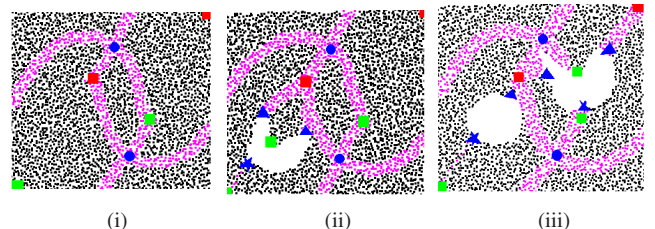


Fig. 7. Networks used in sweep. Around 4000 nodes, UDG, average node degree ~ 17 . (i) without any holes; (ii) With One hole; (iii) With two holes.

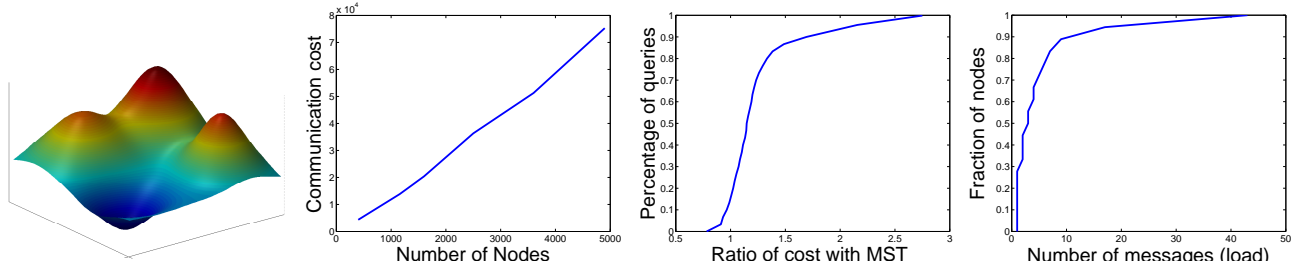


Fig. 6. (i) The continuous signal sampled by sensors; (ii) Message complexity of Morse-Smale decomposition; (iii) CDF of ratio of query cost to cost of MST (iv) CDF of node load distribution.

with collisions and exponential back-off was used in these experiments. Time was measured by ticks of a global clock. We measured the total time taken to complete the sweep, average and max up-time per node, and the total number of medium access collisions. The up-time of a node was measured as starting at the tick when it received its first invitation to join the sweep, to the time it successfully sent out the final invitation to a lower neighbor. It measures how long a node needs to stay active during the sweep and correlates to its energy consumption. The results for the three networks above are presented in tables I, II and III respectively. We compare the performance of the sweep on the decomposition with that of the sweep without the decomposition, and with that of the sweep by a geographic coordinate (say X coordinate). The signal function is the one shown in figures 2(i) & (ii).

Type	Total	Avg up-time	Max up-time	#collisions
Normal	12444	563.5	4376	9208
X coord	8602	311.3	992	8609
Decomposed	6927	288.0	1260	5056

TABLE I. The sweep time for the network without hole in fig 7(i).

Type	Total Time	Avg up-time	Max up-time	#collisions
Normal	12495	636.3	5893	8853
X coord	9060	335.3	2700	8381
Decomposed	6416	273.4	1591	5387

TABLE II. The sweep time for the network with 1 hole in fig 7(ii).

Type	Total Time	Avg up-time	Max up-time	#collisions
Normal	17182	984.5	12147	10013
X coord	11748	579.1	8475	9156
Decomposed	7788	386.1	2056	5766

TABLE III. The sweep time for the network with 2 holes in fig 7(iii).

It is easy to see that the sweep on the decomposition outperforms all other schemes in almost all respects. This is to be expected, since the partitioning allows for more parallelism and less collisions. It avoids sweeps meeting at saddle points, and thus prevents a fast moving sweep frontier piece from waiting a long time for other contour pieces to catch up. The effect is even more pronounced in complex network topologies with holes. See table III where the maximum up-time of a node is many times smaller in the sweep on the decomposition. As explained in Fig. 4 and related discussion, a regular frontier sweep may have to wait a very long time to merge with a companion frontier. This waiting time is eliminated in the sweep on the decomposition.

V. CONCLUSION

We developed a distributed algorithm to decompose a sensor field with respect to the sensor data so that each cell is simply

connected with a homogeneous data flow. The philosophy of exploiting the sensor data characters in implementing networking operations is a promising direction in sensor network research.

REFERENCES

- [1] M. D. Carmo. *Riemannian Geometry*. Birkhäuser, Boston, Basel, Berlin, 1992.
- [2] H. Edelsbrunner, J. Harer, and A. Zomorodian. Hierarchical Morse complexes for piecewise linear 2-manifolds. In *Proc. 17th Ann. ACM Sympos. Comput. Geom.*, pages 70–79, 2001.
- [3] Q. Fang, J. Gao, L. Guibas, V. de Silva, and L. Zhang. GLIDER: Gradient landmark-based distributed routing for sensor networks. In *Proc. of the 24th Conference of the IEEE Communication Society (INFOCOM)*, volume 1, pages 339–350, March 2005.
- [4] R. Forman. A discrete morse theory for cell complexes. In S. T. Yau, editor, *Geometry, Topology and Physics for Raoul Bott*. International Press, 1995.
- [5] R. Forman. Morse theory for cell complexes. *Advances in Mathematics*, 134:90–145, 1998.
- [6] S. Funke and C. Klein. Hole detection or: “how much geometry hides in connectivity?”. In *SCG '06: Proceedings of the twenty-second annual symposium on Computational geometry*, pages 377–385, 2006.
- [7] J. Gao, L. J. Guibas, S. Y. Oudot, and Y. Wang. Geodesic delaunay triangulation and witness complex in the plane. In *Proc. of ACM-SIAM Symposium on Discrete Algorithms (SODA'08)*, pages 571–580, January 2008.
- [8] J. Gao, R. Sarkar, and X. Zhu. Morse-smale decomposition, cut locus and applications in wireless sensor networks. <http://www.cs.sunysb.edu/~jgao/paper/morse.pdf>, 2008.
- [9] R. Ghrist and A. Muhammad. Coverage and hole-detection in sensor networks via homology. In *Proc. the 4th International Symposium on Information Processing in Sensor Networks (IPSN'05)*, pages 254–260, 2005.
- [10] A. Kröllner, S. P. Fekete, D. Pfisterer, and S. Fischer. Deterministic boundary recognition and topology extraction for large sensor networks. In *Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1000–1009, 2006.
- [11] J. W. Milnor. *Morse Theory*. Princeton University Press, Princeton, NJ, 1963.
- [12] 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. of the 27th Annual IEEE Conference on Computer Communications (INFOCOM'08)*, pages 1175–1183, May 2008.
- [13] P. Skraba, Q. Fang, A. Nguyen, and L. Guibas. Sweeps over wireless sensor networks. In *IPSN '06: Proceedings of the fifth international conference on Information processing in sensor networks*, pages 143–151, 2006.
- [14] S. Smale. Morse inequalities for a dynamic system. *Bulletin of the AMS*, 66:43–49, 1960.
- [15] R. Thom. Sur une partition en cellules associée à une fonction sur une variété. *Comptes Rendus de l'Académie de Sciences*, 228:973–975, 1949.
- [16] Y. Wang, J. Gao, and J. S. B. Mitchell. Boundary recognition in sensor networks by topological methods. In *Proc. of the ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)*, pages 122–133, September 2006.