SURF: A Connectivity-based Space Filling Curve Construction Algorithm in High Genus 3D Surface WSNs

Chen Wang Hongbo Jiang
School of Electronic Information and Communications,
Huazhong University of Science and Technology, China, 430074
{cwangwhu, hongbojiang2004}@gmail.com

***Abstract*—Many applications in wireless sensor networks (WSNs) require that sensor observations in a given monitoring area be aggregated in a serial fashion. This demands a routing path to be constructed traversing all sensors in that area, which is also called to linearize the network. In this paper, we present SURF, a Space filling cURve construction scheme for high genus 3D surFace WSNs, yielding a traversal path provably aperiodic (that is, any node is covered at most a constant number of times). SURF first utilizes the hop-count distance function to construct the iso-contour in discrete settings, then it uses the concept of the Reeb graph and the maximum cut set to divide the network into different regions. Finally it conducts a novel serial traversal scheme, enabling the traversal within and between regions. To the best of our knowledge, SURF is the first high genus 3D surface WSNs targeted and pure connectivity-based solution for linearizing the networks. It is fully distributed and highly scalable, requiring a nearly constant storage and communication cost per node in the network. Extensive simulations on several representative networks demonstrate that SURF works well on high genus 3D surface WSNs.**

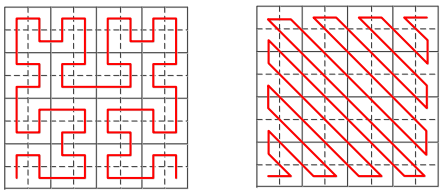
I. Introduction

In this paper we focus on constructing a space filling curve (SFC) to linearize a high genus three-dimensional (3D) surface WSN, i.e., "traversing" the high genus 3D surface network by a single path. In the following, we look back upon related works of the SFC and its applications and constructions in WSNs, offering a full-spectrum understanding toward its advancement in WSNs, followed by our contributions.

A. Existing Work

1) The concept of the SFC: The concept of the SFC came out in the late 19th century, and is accredited to Peano, who proved the existence of a curve that passes through every point of a closed unit square [15]. Thereafter, various SFCs have been proposed for 2D and 3D settings [17]. Most of these curves are recursively constructed, e.g. the Hillbert curve in Fig. 1(a) with respect to three iterations, while some are non-recursive, e.g. the Cantor curve in Fig. 1(b). However, in continuous settings almost all existing SFCs are constructed in a square or cube region, and little work has been done to extend the SFC construction to other shapes, letting alone solutions in discrete settings.

This work was supported in part by the National Natural Science Foundation of China under Grants 61271226, 61272410, and 61202460; by the National Natural Science Foundation of Hubei Province under Grant 2014CFA040; by the Fundamental Research Funds for the Central Universities under Grants 2014QN158, 2014QN164, 2014XJGH003; by the Fok Ying Tung Education Foundation under Grant 132036; by the Hong Kong Scholars Program under Grant XJ2012019; and by the China Postdoctoral Science Foundation under Grants 2013M540580, 2014M560608. The corresponding author of this paper is Hongbo Jiang.



(a) The Hillbert curve. (b) The Cantor curve.

Fig. 1. Two typical SFCs in 2D domain.

2) Applications of the SFC in WSNs: In discrete WSNs, enforcing a linear order of the sensor nodes via the SFC has broad applications, one of which is the serial operation on both sensor nodes and sensor data. In [14], a serial fusion technique was proposed for collaborative signal detection by the traversal along a SFC, which ensures visiting the nodes in a linear order. This technique enables the detection process to stop as soon as sufficient evidences have been collected. In [20] the authors presented a dual addressing space representation architecture for self-organizing networks, and in [5], the authors proposed an algorithm for processing similarity search queries in WSNs. They both applied the Hilbert curve to linearize the network, establishing a bijective mapping between the SFC and the node's index/logical address, so that standard 1D indexing/addressing mechanisms can be applied.

Another application of the SFC is the motion planning of mobile beacons. [2] and [11] considered the localization of WSNs, where a single mobile beacon aware of its position was utilized to help other nodes to localize themselves by moving to cover the entire network. The paths that the mobile beacon should travel along are designed to follow the SFCs. Similar ideas can also be helpful for the battery recharge of the sensors [22] or the data collection by the data mules near the sink [13], [19].

3) The SFC construction in WSNs: The previous works mainly focus on applying the SFC for various purposes in WSNs. How to construct the SFCs, however, is not their concentration. The only research concentrates on constructing the SFCs in WSNs is presented in [3]. The proposed method can generate a SFC that densely covers any planar geometric domain, with a coverage density proportional to the length of the curve, but it cannot be directly utilized in 3D scenarios.

978-1-4799-8381-0/15/$31.00 ©2015 IEEE

A practically potential solution for visiting nodes in a 3D network could be based on the random walk. The problem, however, is that when choosing its next hop, the random walk is essentially "blind". Intuitively, a random walk visits a new node with high probability at the initial stage. But after a certain proportion of nodes have been visited, the random walk is more likely to take a (infinite) long period to aimlessly walk in the network, anchoring its hope on encountering the last few unvisited nodes earlier. Therefore, the random walk cannot ensure deterministic node covering results with restricted traversal path length.

4) High Genus 3D Surface WSNs: Although 2D planar and simple 3D volume settings are assumed in most earlier WSN studies, there have been increasing interests in scenarios where sensors are typically deployed in complex-connected 3D surfaces. Examples of such applications include the fire prevention in the corridors of buildings [8], the monitoring of coal mine tunnels for disaster warning [12], as well as the monitoring in underground tunnels used in gas, water or sewer systems [1]. This kind of WSNs is often of a complex-connected 3D setting with non-trivial topology, being modeled as high genus[1] 3D surface WSNs. It comes to our attentiones that only a few of routing studies [23], [24] are conducted on these emerging networks.

It is noted that, the task of SFC construction is significantly different from packet routing path designs in high genus 3D surface WSNs that have been addressed in [23], [24]. Rather than connecting two given nodes by a routing path, the goal of SFC construction is a path traversing all sensor nodes (a coverage-critical selection). This task is more challenging, considering the less-studied high genus 3D surface WSNs.

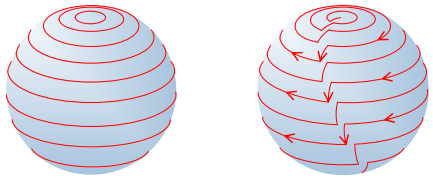
B. Challenges and Our Contributions

In this paper, we present SURF, a novel Space filling cURve construcion scheme for high genus 3D surFace WSNs, yielding a path provably aperiodic (any node in the path is covered/visited at most a constant number of times). The name of our proposed scheme, SURF, can be interpreted as the surface filling or surfing on 3D surfaces.

The intuition of SURF stems from the observation in the continuous domain — the iso-contours of a closed spherical (genus-0) surface in the smooth setting naturally form an embryonic form of the SFC (Fig. 2(a)), and directly connecting them forms a SFC shown in Fig. 2(b). When it comes to high genus surfaces, the primary challenge lies in the way how to deal with the genuses, which may incur two or more contours of an iso-value. Fig. 4(a) shows a case with only one contour of the iso-value 1, but two contours of the iso-value 10.

The main idea behind SURF is to divide the surface into a set of genus-0 regions by cutting off the genuses, followed by constructing the SFC in each region separately before connecting all SFCs in a practical way. The particular problem we are facing in discrete networks, however, is more challenging: (1)

[1]The genus is a topological term that can be simply regarded as the number of handles on the sphere, see Section II-A. Without leading to confusion, we interchangeably use the terms "genus" and "handle".



(a) (b)

Fig. 2. (a) Iso-contours of a shpere. (b) A SFC of a sphere in 3D domain where iso-contours are connected one by one.

How to define the iso-contour in a discrete 3D surface network with mere connectivity information, such that the iso-contour is a connected closed curve? (2) How to identify genuses and further cut them off to form different regions? (3) How to ensure the length of the generated SFC is bounded?

To address the first challenge, we propose to use the hop-count distance function to construct the iso-contour in discrete settings (detailed in Section III-A). For the second one, the concept of the Reeb graph and the maximum cut set are utilized to realize the network segmentation (detailed in Section III-B). After these procedures, the network is divided into different regions. We design schemes to guarantee the traversal within and between regions, yielding a SFC provably aperiodic (see Theorem 5 in Section III-C). The whole process, as an example, is shown in Fig. 3.

To the best of our knowledge, SURF is the first high genus 3D surface WSNs targeted and pure connectivity-based solution for the SFC construction in WSNs. SURF offers several salient features. First, it requires connectivity information only, without the reliance on the location or distance measurement. Second, it does not rely on any particular communication model, only assuming a constant maximum transmission range, which is a common case in practical WSNs. Third, it is fully distributed and scalable, with a nearly constant storage and communication cost of every node.

The rest of this paper is organized as follows. In Section II, we introduce some preliminary knowledge of our proposed SURF algorithm. In Section III, we describe SURF in detail and analyse its storage and communication cost. Section IV presents the performance evaluation, and finally, Section V concludes the paper.

II. Preliminary

Before digging into the problem of SFC construction, we briefly introduce several notions and definitions in algebraic topology and computing geometry. For further theoretical background, we refer interested readers to [10].

A. Cut and Genus

In algebraic topology, a *cut* C is referred to as a disjoint closed simple curve on a connected and orientable surface $\mathcal{M}$, where $\mathcal{M}$ is orientable, indicating it has two distinct sides. One notable property of a cut is its ability to *locally* disconnect the topology of $\mathcal{M}$. Suppose a set $\mathcal{C} = \{C_1, C_2, ..., C_n\}$ is a cut set on $\mathcal{M}$, whose cardinality is n, then $\mathcal{C}$ is a *maximum cut set*

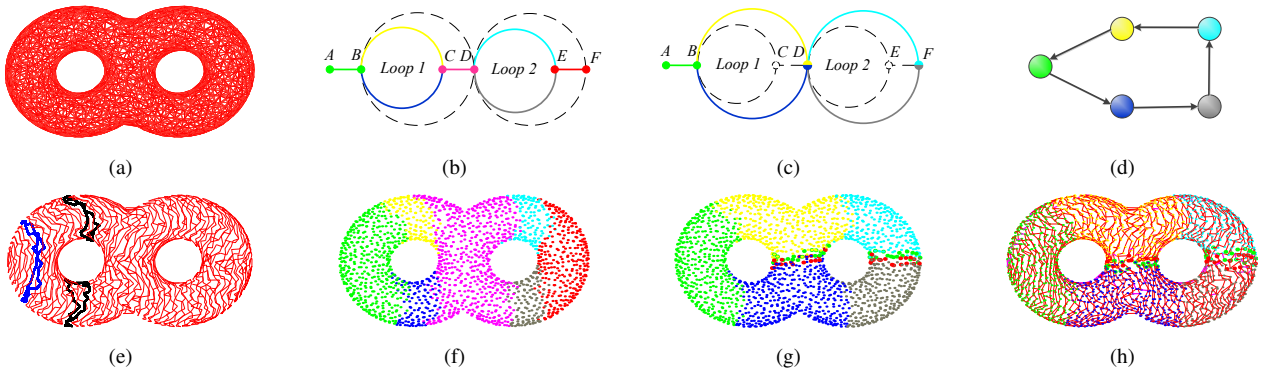


Fig. 3. The pipeline of SURF. (a) The triangulated genus-2 torus network. (b) The Reeb graph of the network. (c) The Reeb graph after the bisection operation. (d) The traversal sequence of SURF in different regions. (e) The iso-distance contour lines of the network. (f) The regions of the Reeb graph in (b). (g) The regions of the Reeb graph in (c); the cut pairs are represented by red and green dots. (h) The SFC generated by SURF, with link edges colored in pink. Different arcs and regions of the Reeb graph are distinguished by colors.

$\mathcal{C}_{max}$ of $\mathcal{M}$ if and only if: (1) Any two cuts in $\mathcal{C}_{\max}$ belong to different homotopy classes, i.e., one cut cannot be smoothly deformed to another without leaving the surface; (2) $\mathcal{M} \backslash (C_1 \cup C_2 \cup ... \cup C_n)$ is connected; (3) $\mathcal{M} \setminus (C_1 \cup C_2 \cup ... \cup C_{n+1})$ is disconnected. Accordingly, the *genus* of $\mathcal{M}$ is defined as the cardinality of the $\mathcal{C}_{max}$ of $\mathcal{M}$, indicating the maximum number of cuts without rendering $\mathcal{M}$ disconnected [7].

The notion of the genus is closely associated with the classification of orientable closed surfaces up to homeomorphism: for a given integer $n \geq 0$, there is exactly one topological type, "the surface of genus-n", which can be obtained by attaching n handles onto a simple closed 3D surface. For example, a sphere is a genus-0 surface, and any cut will render it disconnected; a torus is a genus-1 surface, as shown in Fig. 4, with at most one cut on it while not leading to its disconnection.

Motivated by [23] where the surface $\mathcal{M}$ is decomposed to a simple genus-0 topology using cuts, our idea is to exploit a similar method to generate a sliced surface with which SURF is able to extract the iso-distance contour and the Reeb graph, and we hereby put forward a novel serial traversal scheme for the SFC construction.

B. Iso-distance Contour

For a scalar, real-valued function $f : \mathcal{M} \to \mathbb{R}$, the *level set* of an *iso-value* h is the set of points $f^{-1}(h) = \{p \in \mathcal{M} \mid f(p) = h\}$ [4]. A *contour* is a connected component of a level set, i.e. a curve along which f has a constant value. Fig. 4(a) illustrates a set of successive iso-distance contours on a torus with a mapping to an integer set $\{0, 1, ..., H\}$ of different height values.

In a discrete network with mere connectivity information, however, it is not straightforward to define the level set where no height value or geodesic distance[2] can be derived. What's more, even if we can find a metric to represent the level set, it is still unclear whether there is a connected component to form the contour.

[2]The geodesic distance on a 3D surface can be regarded as the (locally) shortest path between two nodes on the surface.

To tackle this problem, SURF makes use of the hop-count distance, an analogy of the Euclidean/geodesic distance in continuous domains, to define the real-valued function f. Meanwhile, SURF is designed to ensure the existence of the contours (detailed in Section III-A). Note that one challenge here is, when a level set is separated to two or more contours due to the existence of the genuses (e.g. $f^{-1}(10)$ in Fig. 4(a)), SURF is supposed to have the ability to find out where the genuses are. For this purpose, the Reeb graph is used to extract a maximum cut set that cuts off the genuses of $\mathcal{M}$.

C. Reeb Graph and Cut Identification

Reeb graph is a topological structure proposed in [16]. Briefly speaking, a Reeb graph $\mathcal{R}$ of a real-valued function f explicitly reveals the evolution of its level set $f^{-1}(\cdot)$. When the number of the contours of $f^{-1}(\cdot)$ increases or decreases, the gradient of f will vanish at the separating points of the contours. Those points are called *critical points* (theoretically, there are three types of critical points, namely, *minima*, *saddles*, and *maxima* [16].) of f, e.g., Saddle A and Saddle B in Fig. 4(b) which shows an example of the Reeb graph.

Given a Reeb graph, we turn to extracting a maximum cut set $\mathcal{C}_{\max}$ from $\mathcal{M}$. Our approach is motivated by the following theorem.

Theorem 1: The Reeb graph of a closed orientable genus-n 2-manifold has exactly n loops [6].

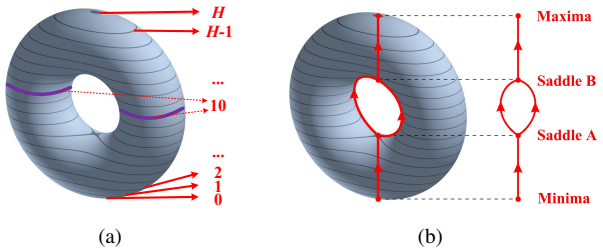


Fig. 4. Iso-distance function and its Reeb graph on a torus.

Theorem 1 implies that we can first identify all loops of the Reeb graph, thereby finding a cut for each loop. Specifically, a loop in a Reeb graph is associated with two degree-3 nodes: one starts the loop and the other ends it, see the two saddles in Fig. 4(b). Thus, we have

Definition 1: An arc of the Reeb graph of $\mathcal{M}$ is a *loop-end arc*, if it is merged from two different arcs.

See Fig. 3(b) for an example where the blue and yellow arcs are merged into a loop-end one colored in pink.

Corollary 2: Each loop in the Reeb graph of $\mathcal{M}$ corresponds to one loop-end arc.

As such, in order to identify a cut for one loop, our method is to find the bisection in loop-end arc which disconnects this loop. Fig. 3(c) shows the Reeb graph after the bisection operation. We will present the implementation in a discrete network in Section III-B.

III. SURF Algorithm

Given a high genus 3D surface WSN, SURF is derived from its triangular form. However, the triangulation procedure itself is out of the scope of this work. Numerous recent studies, e.g. [9], [18], [25], have proposed simple and distributed algorithms to obtain the triangular structure, and can be used in conjunction with our approach. The triangular structure offers a shape representation of the high genus 3D surface, as shown in Fig. 3(a). Without leading to confusion, hereafter we still refer to this triangular structure as the high genus 3D surface, denoted by $\mathcal{M}$, with its vertex (node) set $\mathcal{V} = \{v_i\}$ and edge set $\mathcal{E} = \{e_{ij} = (v_i, v_j) \mid v_j$ is called the *neighbor* of $v_i\}$. Given a triangular structure, SURF follows three steps for the SFC construction:

(1) *Contour Construction*: to lay the groundwork for regional division (see Fig. 3(e)).

(2) *Maximum Cut Set Identification*: to cut off the genuses and thus divide the network to different regions (see Figures. 3(g)).

(3) *Serial Traversal Scheme*: to finally construct the SFC by the traversal intra and inter-regions (see Fig. 3(h)).

A. Contour Construction

The first step of SURF is to construct the level set and its corresponding contour lines[3] of $\mathcal{M}$, so that, in each contour line, it is trivial to locally construct a SFC. To that end, we first establish a hop count distance function $f : \mathcal{M} \to L$, where L is the integer set representing the hop count. Specifically, a randomly selected root node r initiates a flooding across the whole network. After receiving a flooded message from r, every node knows its hop count distance l to r, and then records its level index with l. Therefore, for any node v_i with the level index l, we have $f(v_i) = l$. Accordingly, the *level set* of hop count distance l is given by $f^{-1}(l) = \{e_{ij} = (v_i, v_j) \in \mathcal{E} \mid f(v_i) = f(v_j) = l\}$.

Recall that a contour in continuous settings is a connected component of a level set. In the following, we show that in discrete networks, a connected contour line of a level set also exits, which is defined based on the following notion.

Definition 2: The *l-neighbor* for any edge e_{ij} in $f^{-1}(l)$ is defined as $N(e_{ij}, l) = \{(e_{ij}, e_{ik}) \mid e_{ij}$ and e_{ik} have a common vertex $v_i \in \mathcal{V}$, and $e_{ij}, e_{ik} \in f^{-1}(l)\}$.

Then we introduce the concept of iso-distance contour line and its property in discrete settings.

Definition 3: An *iso-distance contour line* (*iso-contour* for short) of $f^{-1}(l)$, $\mathcal{O}(l)$, is defined as a neighbor graph of $f^{-1}(l)$, such that:

(1) $\bigcap_{\forall \mathcal{O}(l)} \mathcal{O}(l) = \varnothing$; and (2) $\bigcup_{\forall \mathcal{O}(l)} \mathcal{O}(l) = \bigcup_{e_{ij} \in \mathcal{E}} N(e_{ij}, l)$.

As the iso-contour of $f^{-1}(l)$ is defined as the neighbor graph of $f^{-1}(l)$, we have the following Lemma.

Lemma 3: Any two nodes in an iso-contour of $f^{-1}(l)$ is l-connected, where two nodes in $f^{-1}(l)$ is *l-connected*, if between them there is a path, the nodes on which are all in $f^{-1}(l)$.

Lemma 3 implies the connectivity of an iso-contour, while no guarantee of its closeness. In fact, there exist special nodes, so-called "dead-end", rendering the iso-contour unclosed. They are closely related to the following parent/child relationship.

Definition 4: For any $e_{pc} = (v_p, v_c) \in \mathcal{E}$, if $f(v_p) = l$, $f(v_c) = l + 1$, then v_p is a *parent* node of v_c; v_c is a *child* node of v_p.

Definition 5: A node is a *dead-end*, if it has no child nodes. Correspondingly, an edge $e_{ij} = (v_i, v_j)$ is a *dead-end edge* if either v_i or v_j is a dead-end.

We then have the following theorem that guarantees the closeness of an iso-contour.

Theorem 4: An iso-contour with dead-end edges eliminated is a connected and closed cycle.

Proof: See the Appendix. ∎

So far, it is trivial to establish the discrete counterpart of a continuous contour: to treat every dead-end edge as a double-edge (detailed in Section III-C). For high genus surfaces where there exist two or more iso-contours in a level set (see the two iso-contours colored in black of $f^{-1}(14)$ in Fig. 3(e)), we utilize the Reeb graph to cut off the genuses, and thus divide the network into regions. Further strategies are then proposed to guarantee the traversal intra and inter-regions.

B. Maximum Cut Set Identification

Based on the hop count distance function, we next use the Reeb graph to identify the maximum cut set of $\mathcal{M}$. To that end, a distributed algorithm similar to that in [23] is carried out, which evolves four major sub-steps:

The first sub-step is to identify nodes in each iso-contour of $f^{-1}(l)$ with an iso-contour ID. This is done by randomly selecting one landmark (so-called g-landmark) in an iso-contour. After the g-landmark is selected, it performs flooding within $f^{-1}(l)$, with the messages containing its iso-contour ID and level index l. As such, all nodes in the iso-contour of $f^{-1}(l)$ have the knowledge of the iso-contour ID.

Secondly, all the iso-contours in $\mathcal{M}$ are composed to *regions* (arcs) of the Reeb graph. We say two iso-contours $\mathcal{O}(l)$ and

[3]We use the "contour line" in discrete network settings to distinguish it from the "contour" in continuous scenarios.

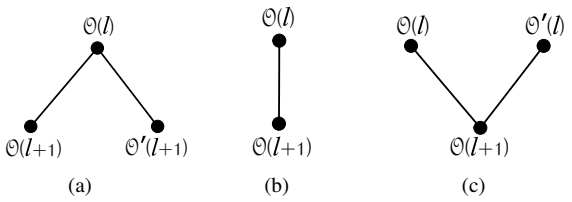


Fig. 5. (a) The iso-contour $\mathcal{O}(l)$ is connected with $\mathcal{O}(l+1)$ and $\mathcal{O}'(l+1)$. (b) The iso-contour $\mathcal{O}(l)$ is connected with $\mathcal{O}(l+1)$ only. (c) The iso-contour $\mathcal{O}(l+1)$ is connected with $\mathcal{O}(l)$ and $\mathcal{O}'(l)$.

$\mathcal{O}(l+1)$ are connected, if there exist a node v in $\mathcal{O}(l)$ that has a neighbor v' in $\mathcal{O}(l+1)$. Then, v and v' will notify this connection to their corresponding g-landmarks. In particularly, there exist three cases related to this connection, as shown in Fig. 5. Correspondingly, the g-landmark notifies all nodes in $\mathcal{O}(l+1)$ if $\mathcal{O}(l+1)$ is only connected with $\mathcal{O}(l)$. In this case, the nodes in $\mathcal{O}(l+1)$ are assigned a region ID the same as that of $\mathcal{O}(l)$; otherwise, the nodes in $\mathcal{O}(l+1)$ will be assigned a new region ID.

With the aforementioned two sub-steps, every node has a region ID. The result of the Reeb graph is shown in Fig. 3(f), where the Reeb graph regions are distinguished in colors.

Next, having the Reeb graph, all loop-end regions (arcs) can be notified directly: if the g-landmark in $\mathcal{O}(l+1)$ is notified that it is connected with $\mathcal{O}(l)$ and another iso-contour $\mathcal{O}'(l)$ (the case in Fig. 5(c)), then nodes in $\mathcal{O}(l+1)$ and all other iso-contours in the same region are notified to be in a loop-end region.

Finally, to extract the maximum cut set $\mathcal{C}_{\max}$, each loop-end region performs a *bisection* operation to extract a cut. Consequently, the loop-end region I_m is bisected and a *merged region* I'_α and I'_β is generated. Then it is simple to identify a cut C_i: each node v in loop-end region I_m sends a message to its neighbor v' in I_m. If v' has a different region ID with v, v and v' are notified to be *cut nodes*. Fig. 3(g) depicts the result of the emerged regions and cut pairs.

C. Serial Traversal Scheme

After cut identification, $\mathcal{M}$ is divided into regions, each of which contains several (uncut or cut) iso-contours. Aiming at a SFC for the whole network, we first conduct the SFC in each iso-contour, thereby connecting those curves for intra and inter-region.

The SFC construction starts from the root node r, which randomly chooses a neighbor as the next hop, say p, and traverses to it. Then p marks itself as visited, and the traversal path $e(r,p)$ becomes the first section of the SFC. After that p finds its next hop node q following NEXTHOP in Algorithm 1, which deals with four situations according to the spatial relationship between p and q.

Situation 1: p and q are within one uncut iso-contour. This situation happens when p is not in the loop-end region, and p has an unvisited neighbor q in the same iso-contour. Recall a simple way to generate a local SFC in an iso-contour by Theorem 4, in this situation, if p is a link node (the first visited node in its iso-contour), the next couple of traversals are all of this situation, until p has no unvisited neighbor in the same iso-contour. And then Situation 3 comes. See Fig. 6(a) for example.

Algorithm 1 NEXTHOP(p)

```
cFlag ← false
rFlag ← false
if p is link node of a cut iso-contour then
    q ← next hop on the shortest path to one randomly chosen cut node in p's iso-contour
else if p is cut node then
    for each l-neighbor ngb of p do
        if ngb is unvisited then
            q ← ngb;  cFlag ← true;  break
    if cFlag = false then
        p follows the shortest path to the c-landmark in p's iso-contour; q ← the c-landmark
else
    for each l-neighbor ngb of p do
        if ngb is unvisited then
            q ← ngb;  cFlag ← true;  break
    if cFlag = false then
        for each unvisited neighbor ngb of p do
            if ngb.level = p.level ± 1 then
                q ← ngb;  rFlag ← true;  break
        if rFlag = false then
            p follows the shortest path to an unvisited r-landmark; q ← the r-landmark
return q
```

Situation 2: p and q are within one cut iso-contour. This situation happens when p is in the loop-end region, and p has an unvisited neighbor q in the same iso-contour. It is noted that a cut iso-contour is part of an uncut iso-contour, and cut nodes are end-points of cut iso-contours. This motivates us to start the traversal from a cut node in a cut iso-contour, and to "exit" the cut iso-contour from a *c-landmark* (a randomly selected node having at least one non-dead-end child). This is done by a local flooding within the cut iso-contour, from each cut node and each c-landmark after cut identification. As a result, each node in the cut iso-contour knows its next-hop node on the shortest path to each cut node and each c-landmark. When p is a link node of a cut iso-contour, it first follows the shortest path pointer to one of the cut nodes in its iso-contour, and then starts traversal as in Situation 1, until p has no unvisited neighbor in the same iso-contour. After that, the last node follows the shortest path pointer to the c-landmark in the iso-contour, and there comes Situation 3. See Fig. 6(b) for instance.

Situation 3: p and q are in two different iso-contours. This situation happens when p has no unvisited neighbor in its iso-contour in Situation 1, or p is a c-landmark without unvisited neighbor in its iso-contour in Situation 2. In the former situation, p will choose an unvisited neighbor in a neighboring iso-contour as its next hop (Note if all the unvisited neighbors of p are dead-ends, p will choose its previous hop node as its next hop, so as to guarantee the link node in the next iso-contour is non-dead-end). In the latter situation, as c-landmark

are selected having at least one non-dead-end child, it directly choose an unvisited neighbor in a neighboring iso-contour as its next hop. If not, then Situation 4 comes. See the link edges connecting two iso-contours colored in pink in Fig. 3(h).

Situation 4: p and q are in two different regions. This situation happens when p has no unvisited neighbor in the same region. To achieve a traversal inter-regions, we use r-*landmarks* for piloting, where the r-landmarks are randomly selected from boundary nodes (not cut nodes and having neighbors in another region) in the same region. The r-landmark selection process can be done in a region similar to g-landmark selection in an iso-contour. As a result, each region has one or more r-landmarks (depending on how many regions it borders upon). Next, each r-landmark in one region initiates a flooding, so as to know the next-hop node on the shortest path to those r-landmarks in its neighboring region. Meanwhile, by doing so high-order topological features (i.e. regions and their connections) of $\mathcal{M}$ are identified by r-landmarks. In this situation, p will choose an unvisited r-landmark that has a minimum differential value of level index as its next hop. As an example, Fig. 3(d) shows the traversal order in different regions of the network in Fig. 3(a).

Note that the NEXTHOP algorithm does not consider q as a dead-end. That is because on the one hand, dead-ends of an iso-contour in $f^{-1}(l)$ have no effect on the l-connectivity of other nodes in the iso-contour (by Lemma 7); and on the other hand, in our scheme several methods are utilized to ensure the link nodes are not dead-ends. So before p finds its next node by the NEXTHOP algorithm, it first checks whether it has an unvisited dead-end neighbor d, if does so, it just goes to d, comes back, and continues to seek for its next hop; d marks itself visited. See Fig. 6(c).

The path stretches as the serial traversal goes on, and in the end, a SFC of $\mathcal{M}$ is constructed. Fig. 3(h) shows the conducted SFC of the genus-2 torus network in Fig. 3(a). From the aforementioned serial traversal scheme, it is not hard to arrive at the following theorem that ensures the validity and feasibility of SURF.

Theorem 5: The SFC generated by SURF ensures that every node in $\mathcal{M}$ is covered at most $(\max_{v_i \in \mathcal{V}} n_d(v_i) + n_r + 2)$ times, where n_r is the number of the regions in the network, and

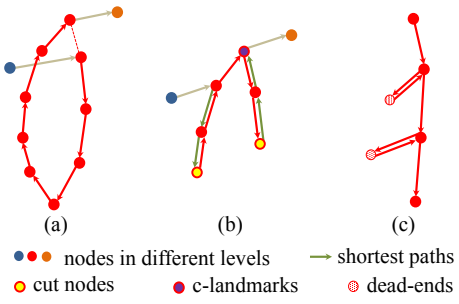


Fig. 6. Several cases in serial traversal. (a) The traversal in an uncut iso-contour. (b) The traversal in a cut iso-contour. (c) The traversal in an iso-contour with dead-ends.

$n_d(v_i)$ is the number of v_i's dead-end neighbors.

Proof: Without regard to the dead-ends, we consider the following four situations: the traversal 1) within an uncut iso-contour, 2) within a cut iso-contour, 3) between two consecutive iso-contours within one region, and 4) between two regions. First, the traversal within an uncut iso-contour generates a local SFC based on Theorem 4, and every node is covered only once. Second, the traversal within a cut iso-contour results in a local SFC, owing to the cut node and the c-landmark as the "entrance" and "exit" indicator in the cut iso-contour, and every node is covered at most 3 times, e.g. the c-landmark if it is also the link node of the iso-contour. Third, two consecutive local SFCs within one region are connected by the link edge, where the link node is designed non-dead-end, and thus the connection is guaranteed; note that the covered times of the nodes associated with link edges are counted in the traversal within their respective cut/uncut iso-contours. Finally, the SFCs in different regions are connected by the r-landmarks in a sorted order, and there may be some node in $\mathcal{M}$ to be covered $(n_r - 1)$ times, in the worst case, if it is shared by all the traversals inter-regions. As the whole process is well-guided in a deterministic manner, every node in $\mathcal{M}$ is covered at least once, and at most $(n_r + 2)$ times. When taking into account the dead-ends, the upper bound of the node's covered times becomes $(\max_{v_i \in \mathcal{V}} n_d(v_i) + n_r + 2)$, in that the current node will just conduct a back-and-forth travel to every unvisited dead-end neighbor. Thus far, Theorem 5 holds. ■

D. Storage and Communication Cost

Storage and communication cost for nodes in WSNs are important factors concerning scalability for a SFC construction algorithm. Here, the storage cost is measured by the number of the nodes (landmarks or IDs) stored, and the communication cost is measured by the number of messages exchanged. We shows the scalability of SURF by

Theorem 6: Both the storage cost and the communication cost of every node in SURF are at most $O(n_r)$, where n_r is the number of r-landmarks in the network.

Proof: Let's see the storage cost at first. First, every node in the network maintains its neighbors' ID with $O(1)$ storage. Second, every node in the cut iso-contours maintains a routing table pointer to every cut node and every c-landmark in its iso-contour, which incurs $O(1)$ storage cost. Third, every r-landmark maintains a routing table pointer to every other r-landmark, with the storage cost $O(n_r)$. Hence, the storage cost of every node in SURF is at most $O(n_r)$.

Next, we turn to communication cost. First, to construct the iso-contours, every node has to communicate to its neighbors to carry out the flooding initiated by the root node, and thus the communication cost is $O(1)$. Second, during the Reeb graph establishment, every node in the process of g-landmark selection, iso-contour ID notification and region division generates $O(1)$ communication cost, respectively. Third, every node in the process of cut identification has $O(1)$ communication cost. Finally, in the serial traversal process, the communication cost is dominated by the traversal among different regions, which

incurs $O(n_r)$ communication cost. Overall, the communication cost of every node in SURF is at most $O(n_r)$. ■

Often in comparison with the network size, the number of r-landmarks is considered negligible. Here it is noted that we do not consider severe cases such as a narrow line-like network with many tiny genuses.

E. Discussion

One may concern that the constructed SFC does not have an adaptive density, while causing a nearly fixed length path. In some cases, it is undesirable to traverse an entire region before getting information from another region. What is more, in such applications in WSNs as the serial fusion and the motion planning of mobile beacons, the length of a path may be restricted by the travel budget or the required fusion delay [3].

To tackle this problem, SURF can be slightly modified, obtaining a "sparser " SFC to meet a given travel budget (that could be much smaller than the length of the entire curve generated by SURF). One simple and feasible solution is to construct a SFC not by connecting every iso-contour. Instead, we connect every kth iso-contour, where k is a parameter adapting to the budget. Then the generated SFC should have a shorter length (hopefully close to one kth). By doing so, one can quickly tour around the network coarsely, get a rough idea of the sensor data and gradually refine the density when more travel budget is available or higher delay is allowed.

IV. Performance Evaluation

To evaluate the effectiveness of SURF, we have conducted extensive simulations on various scenarios. We first examine the performance of SURF in four 3D surface networks: a genus-1 corridor, a genus-2 bowknot, a genus-3 smile and a genus-4 window, with the average node degree between $8 \sim 11$. Fig. 7 shows the results of SURF, as well as the statistical distribution of nodes' covered times. It is observed that, SURF delivers a stable performance and successfully generates in each network a SFC, which traverses the whole network with each node covered a small number of times. This is consistent with Theorem 5.

We also evaluate another two SFC construction approaches for comparison. Note that there are very limited algorithms to compare with because previous SFC construction schemes are proposed for 2D WSNs and cannot be directly applied in high genus 3D surfaces. Among them, random walk may be the sole comparable scheme. However, random walk either does not ensure a full coverage, or results in a long traversal path length, as discussed in Section I-A3. Hence, we additionally propose random-SURF, a variant of SURF, as an alternative comparison object. Random-SURF makes traversal first in different iso-contours of one region, and then in different regions, as is done in SURF. The difference is, random-SURF chooses its next hop (intra/inter regions) at random.

Fig. 8 illustrates the network coverage percentage of random walk, random-SURF and SURF in each 3D network at the first time the network is fully covered by SURF. We can see that SURF yields a faster traversal speed than other two

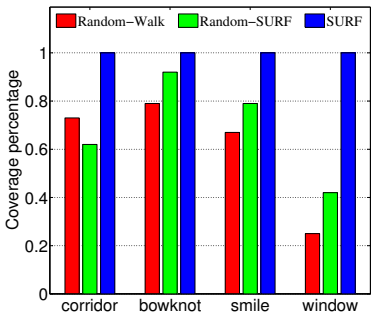
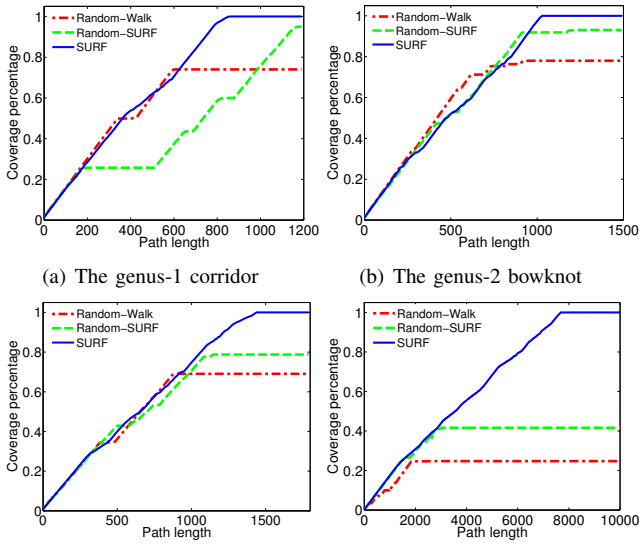


Fig. 8. Comparison on the network coverage.



(a) The genus-1 corridor (b) The genus-2 bowknot

(c) The genus-3 smile (d) The genus-4 window

Fig. 9. The coverage v.s. the path length.

schemes, and the superiority of SURF is even more prominent as the network becomes complex, say in the network of genus-4 window. The better performance of SURF benefits from its region division as well as c-landmark/r-landmark piloting, while random walk and random-SURF are unrefined schemes without explicit directions on how to speedily and orderly traversing the whole network.

Fig. 9 further depicts how the specific network coverage changes as the generated path moves forward by the three algorithms in different networks. It can be seen that, SURF always leads to a 100% network coverage however complex the network is, and also with the fastest pace, while random walk cannot achieve a full coverage in an competing pace. This is not much surprising as SURF is a well-guided algorithm while random walk is somewhat blind and luck-dependent.

It is also found that, in comparison with random walk, random-SURF performs better in simpler topologies, say the networks of genus-2 bowknot and genus-3 smile. The reason is that, in contrast to random walk's complete blindness, random-SURF traverses the network in a more global way: before aimlessly choosing its next hop, it has to find an unvisited neighbor in its iso-contour or in its region, which to some

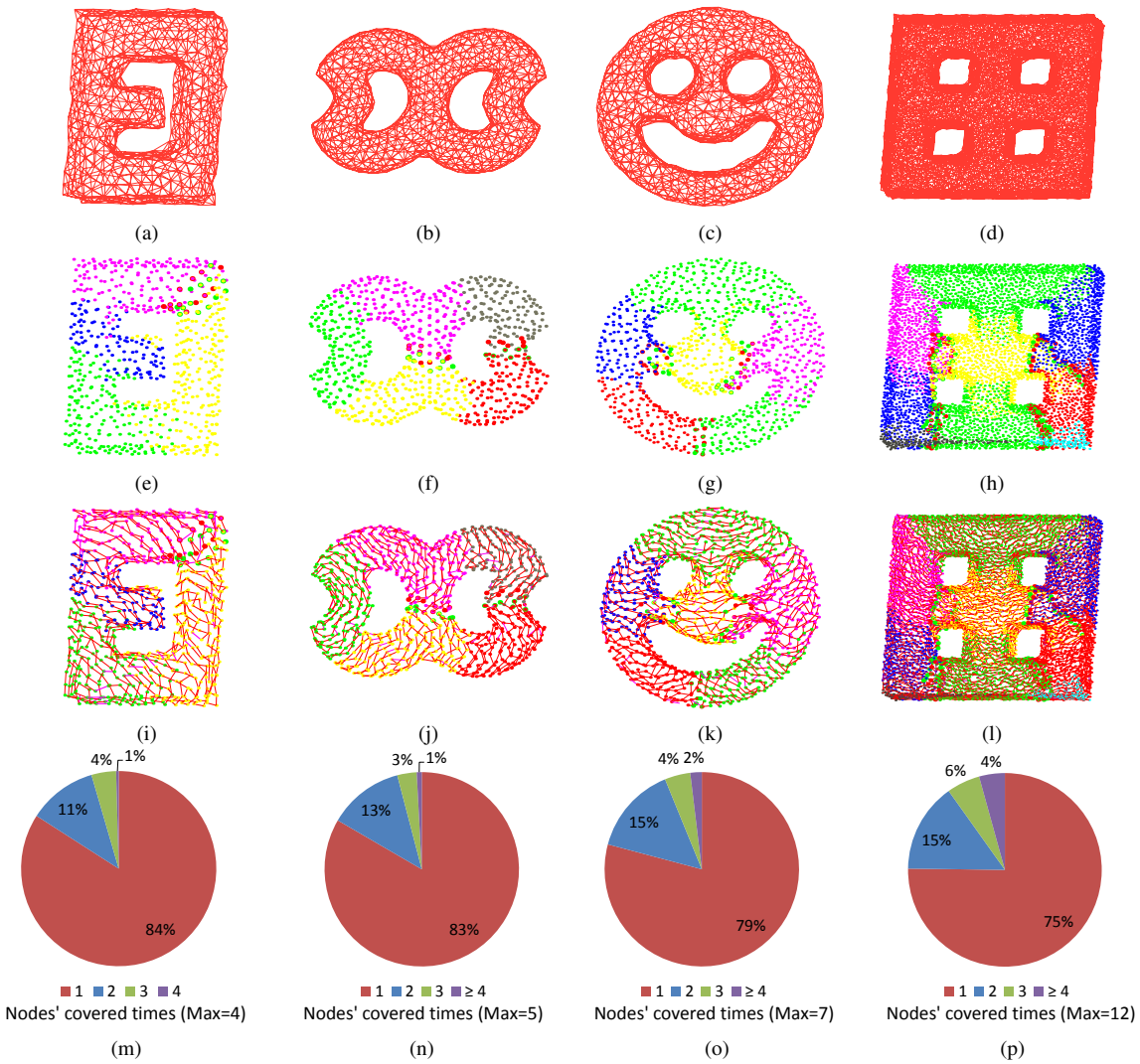


Fig. 7. Columns from left to right: (a) A genus-1 corridor network with 710 nodes; Avg deg is 8.92. (b) A genus-2 bowknot network with 837 nodes; Avg deg is 9.35. (c) A genus-3 smile network with 1,102 nodes; Avg deg is 10.01. (d) A genus-4 window network with 5,366 nodes; Avg deg is 9.63. Rows: (1) The original triangulated network. (2) The emerged regions of the Reeb graph and the maximum cut set. (3) The SFC generated by SURF. (4) The distribution of nodes' covered times.

extent alleviates its blindness.

In more complex scenarios, e.g. in the genus-4 window, however, random-SURF is still far less effective than SURF with respect to both coverage rate and coverage speed. After all, random-SURF is randomized. Without the guidance of c-landmarks in cut iso-contours or g-landmarks inter-regions, random-SURF is inevitable running into locally infinite loop at a certain time. That is why in Fig. 9(d), the coverage percentage of random-SURF nearly keeps steady when the path length is greater than roughly 3000.

As a matter of fact, in our tests in the genus-4 window, either random walk or random-SURF requires a path length more than 50 thousands to reach a 50% coverage. This is because after a certain fraction of nodes have been visited, both random walk and random-SURF are more inclined to aimlessly find the last few unvisited nodes with a long time, as we discussed in Section I-A3.

V. Conclusion

We have presented a novel distributed algorithm for SFC construction in high genus 3D surface WSNs. It requests the connectivity information only, and does not require in advance knowledge of location or distance information. It is also scalable since the node's storage and communication cost are independent on the network size. It has been proved that our algorithm can generate a path covering each node at most a constant number of times. Extensive simulations demonstrate the effectiveness of our algorithm.

In the future, we would like to consider how to realize a proportional coverage of the generated SFC, with an adaptive density for a given traversal budget or delay constrain as in [3], [21]. Besides, we only consider in this paper the orientable closed 3D surface (compact and without boundaries). It would be more challenging to design linearization schemes for more general 3D surface networks, say with holes.

APPENDIX
PROOF OF THEOREM 4

A. Lemma 7

Lemma 7: Dead-ends of an iso-contour in $f^{-1}(l)$ have no effect on the l-connectivity of the non-dead-end nodes within the same iso-contour.

Proof: According to the size of the l-degree (defined as the number of a node's neighbors in its iso-contour), dead-ends can be classed to the following two categories: dead-ends with l-degree one, and dead-ends with l-degree more than one. Fig. 10 shows the two cases of dead-ends.

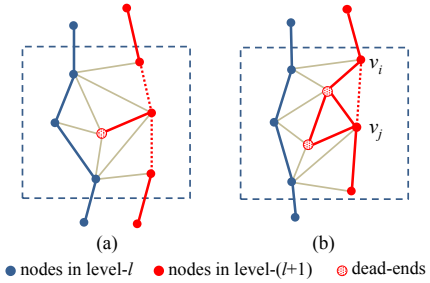


Fig. 10. Two cases of dead-ends.

For dead-ends with l-degree one, when they are removed, the l-connectivity of the non-dead-end nodes in the iso-contour is guaranteed by Lemma 3. For dead-ends with l-degree more than one, we first claim that the neighbor nodes of a dead-end in the iso-contour is directly connected (otherwise it contradicts our assumption of a triangulated surface). So in Fig. 10(b), nodes v_i and v_j are directly connected. Then it can be seen that this kind of dead-ends cannot negate the l-connectivity of the non-dead-end nodes in the iso-contour. That is, Lemma 7 holds. ■

B. Proof of Theorem 4

Proof: On the one hand, Lemma 3 guarantees that any two nodes in the same iso-contour are connected, and from Lemma 7 we can see that dead-ends have no effects on the connectivity of the iso-contour. So an iso-contour with dead-end edges eliminated is connected.

On the other hand, we prove that an iso-contour with dead-end edges eliminated is a closed cycle. That can be done by the mathematical induction. We start from the root node $r = f^{-1}(0)$. Recall that the network is a closed surface, so $f^{-1}(1)$ contains one or more closed curves, i.e., any one of the iso-contours of $f^{-1}(1)$ is a closed cycle. Note that there is no dead-end edge in the iso-contour(s) of $f^{-1}(1)$. Assume any one of the iso-contours of $f^{-1}(l)$ with dead-end edges eliminated is a closed cycle. We next prove that any one of the iso-contours of $f^{-1}(l+1)$ with dead-end edges eliminated is as well a closed cycle. Three cases need to be considered: the iso-contour of $f^{-1}(l+1)$ 1) without dead-ends, 2) with dead-ends of l-degree one, and 3) with dead-ends of l-degree more than one. In all cases, any one of the iso-contours of $f^{-1}(l+1)$ with dead-end edges eliminated is provably a closed cycle following our assumption that $\mathcal{M}$ is closed and triangulated, and we here omit the detail due to space limitations. ■

REFERENCES

[1] M. R. Akhondi, A. Talevski, S. Carlsen, and S. Petersen. Applications of wireless sensor networks in the oil, gas and resources industries. In *Proc. of IEEE AINA*, pages 941–948, 2010.
[2] J. M. Bahi, A. Makhoul, and A. Mostefaoui. Hilbert mobile beacon for localisation and coverage in sensor networks. *International Journal of Systems Science*, 39(11):1081–1094, 2008.
[3] X. Ban, M. Goswami, W. Zeng, X. Gu, and J. Gao. Topology dependent space filling curves for sensor networks and applications. In *Proc. of IEEE INFOCOM*, pages 2166–2174, 2013.
[4] H. Carr, J. Snoeyink, and M. van de Panne. Simplifying flexible isosurfaces using local geometric measures. In *Proc. of IEEE VIS*, pages 497–504, 2004.
[5] Y.-C. Chung, I. Su, and C. Lee. An efficient mechanism for processing similarity search queries in sensor networks. *Information Sciences*, 181(2):284–307, 2011.
[6] K. Cole-McLaughlin, H. Edelsbrunner, J. Harer, V. Natarajan, and V. Pascucci. Loops in Reeb graphs of 2-manifolds. In *Proc. of ACM SoCG*, pages 344–350, 2003.
[7] J. Erickson and S. Har-Peled. Optimally cutting a surface into a disk. *Discrete & Computational Geometry*, 31(1):37–59, 2004.
[8] C. Fischer and H. Gellersen. Location and navigation support for emergency responders: A survey. *IEEE Pervasive Computing*, 9(1):38–47, 2010.
[9] S. Funke and N. Milosavljevic. Network sketching or: 'How much geometry hides in connectivity? – part II'. In *Proc. of ACM-SIAM SODA*, pages 958–967, 2007.
[10] A. Hatcher. *Algebraic topology*. Cambridge University Press, 2002.
[11] D. Koutsonikolas, S. M. Das, and Y. C. Hu. Path planning of mobile landmarks for localization in wireless sensor networks. *Computer Communications*, 30(13):2577–2592, 2007.
[12] M. Li and Y. Liu. Underground coal mine monitoring with wireless sensor networks. *ACM Transactions on Sensor Networks*, 5(2):10, 2009.
[13] J. Luo and J.-P. Hubaux. Joint sink mobility and routing to maximize the lifetime of wireless sensor networks: The case of constrained mobility. *IEEE/ACM Transactions on Networking*, 18(3):871–884, 2010.
[14] S. Patil, S. R. Das, and A. Nasipuri. Serial data fusion using space-filling curves in wireless sensor networks. In *Proc. of IEEE SECON*, pages 182–190, 2004.
[15] G. Peano. Sur une courbe, qui remplit toute une aire plane. *Mathematische Annalen*, 36(1):157–160, 1890.
[16] G. Reeb. Sur les points singuliers d'une forme de Pfaff completement intgrable ou d'une fonction numrique. *Comptes Rendus de L'Acadmie ses Sances, Paris*, 222:847–849, 1946.
[17] H. Sagan. *Space-filling curves*. Springer-Verlag New York, 1994.
[18] R. Sarkar, X. Yin, J. Gao, F. Luo, and X. D. Gu. Greedy routing with guaranteed delivery using Ricci flows. In *Proc. of ACM/IEEE IPSN*, pages 121–132, 2009.
[19] R. Sugihara and R. K. Gupta. Path planning of data mules in sensor networks. *ACM Transactions on Sensor Networks*, 8(1):1–27, 2011.
[20] A. C. Viana, M. Dias de Amorim, Y. Viniotis, S. Fdida, and J. F. De Rezende. Twins: A dual addressing space representation for self-organizing networks. *IEEE Transactions on Parallel and Distributed Systems*, 17(12):1468–1481, 2006.
[21] X. Wang, S. Han, Y. Wu, and X. Wang. Coverage and energy consumption control in mobile heterogeneous wireless sensor networks. *IEEE Transactions on Automatic Control*, 58(4):975–988, 2013.
[22] L. Xie, Y. Shi, Y. T. Hou, and H. D. Sherali. Making sensor networks immortal: An energy-renewal approach with wireless power transfer. *IEEE/ACM Transactions on Networking*, 20(6):1748–1761, 2012.
[23] T. Yu, H. Jiang, G. Tan, C. Wang, C. Tian, and Y. Wu. SINUS: A scalable and distributed routing algorithm with guaranteed delivery for WSNs on high genus 3D surfaces. In *Proc. of IEEE INFOCOM*, pages 2175–2183, 2013.
[24] X. Yu, X. Yin, W. Han, J. Gao, and X. Gu. Scalable routing in 3D high genus sensor networks using graph embedding. In *Proc. of IEEE INFOCOM*, pages 2681–2685, 2012.
[25] H. Zhou, H. Wu, S. Xia, M. Jin, and N. Ding. A distributed triangulation algorithm for wireless sensor networks on 2D and 3D surface. In *Proc. of IEEE INFOCOM*, pages 1053–1061, 2011.