Wireless Netw
DOI 10.1007/s11276-014-0770-6

# GRIP: Greedy Routing through dIstributed Parametrization for guaranteed delivery in WSNs

**Minqi Zhang · Feng Li · Ying He · Juncong Lin · Xianfeng Gu · Jun Luo**



**Abstract** Although stateless greedy routing is well investigated in 2D wireless sensor networks (WSNs), it is widely believed to be impossible in 3D. In this paper, we aim at overcoming the impossibility through a distributed parametrization that equips a WSN with virtual coordinates favoring greedy routing. We propose a fundamentally new parametrization to embed the network domain, the resulting embedding domain allows greedy routing to have guaranteed delivery. We also present localized algorithms to realize this map in WSNs. To combat the load concentration caused by greedy routing that applies the distance greedy principle, we further propose tunable greedy routing, which relies on tuning a parameter in the greedy objective to naturally balance routing load. These two proposals form our *Greedy Routing through dIstributed Parametrization* (GRIP). We prove the correctness and efficiency of GRIP and use simulations to evaluate its performance in terms of complexity, load balancing, and energy efficiency.

**Keywords** Wireless sensor networks · Geographic routing · 3D networks · Distributed parametrization

M. Zhang · F. Li · Y. He · J. Luo (✉)
School of Computer Engineering, Nanyang Technological University, Singapore, Singapore
e-mail: junluo@ntu.edu.sg

M. Zhang
e-mail: mzhang1@ntu.edu.sg

F. Li
e-mail: fli3@ntu.edu.sg

Y. He
e-mail: yhe@ntu.edu.sg

J. Lin
Software School, Xiamen University, Xiamen, China
e-mail: jclin@xmu.edu.cn

X. Gu
Department of Computer Science, State University of New York at Stony Brook, Stony Brook, NY, USA
e-mail: gu@cs.sunysb.edu

## 1 Introduction

Motivated by applications such as underwater surveillance and atmospheric monitoring, the demand in deploying sensor nodes in *three-dimensional* (3D) spaces becomes increasingly keen. Notably, these 3D *wireless sensor networks* (WSNs) do not just occupy a 3D surface; their nodes may instead form 3D volumes. This new feature brings several potential challenges to the protocol designs (e.g., in-network data management [16, 24]). In this paper, we investigate the possibility of enabling greedy routing in 3D WSNs.

Greedy (geographic) routing has long been considered as a scalable data delivery mechanism in large scale multi-hop wireless networks, due to its dependence only on local information [1, 8]. In a nutshell, knowing the geographic location of the destination, the routing decision selects, among all the one-hop neighbors of a forwarding node, the one that is the closest to the destination. Such a protocol is *stateless* as the routing decision is made upon information local to the forwarding node.

Although stateless greedy routing could fall into local minimum due to the existence of *communication voids* (or *holes*) in a network [3], combining it with *face* (or *perimeter*) routing (which relies on a planarized connectivity graph [1, 8] or well detected hole boundaries [3]) may still guarantee data delivery. Unfortunately, as 2D structures such as face or perimeter cannot be trivially extended to

Published online: 09 July 2014

3D, routing decisions relying only on local information cannot guarantee delivery for 3D networks in general [2].

Another body of research works looked at whether stateless greedy routing may work under a certain embedding of a connectivity graph [10, 17]. When exists, such an embedding is termed *greedy embedding*. In particular, Kleinberg [10] shows that every finite connected graph has a greedy embedding in the hyperbolic plane. However, embedding the combinatorial structure of a network requires the embedding to be substantially revamped upon any change in the network, making such a scheme impractical [4].

Our approach in this paper extends the idea of mapping a geometric space rather than embedding a combinatorial structure (e.g., [19]). This is motivated by the fact that, in a densely deployed WSN, the space covered by the network is much more stable than the connectivity graph of the network. While the existing proposals only work for 2D spaces due to their reliance on *conformal structure* (computed by, for example, *Ricci flow* [6]) that exists only on surfaces [19, 22], we propose a *fundamentally new geometric map* valid in both 2D and 3D, and we also propose localized algorithms to perform this map in WSNs.

In terms of performing greedy routing in an embedding domain, existing approaches either stick to the distance greedy principle (leading to an unbalanced load distribution [10, 19]) or rely on complicated transformations to produce multiple routing paths [22]. Our approach, in contrast, only relies on a tunable greedy objective to naturally generate routing path diversity. In summary, our *Greedy Routing through dIstributed Parametrization* (GRIP) has two main components: (1) a localized algorithm to map a network domain $\mathcal{N} \in \mathbb{R}^d, d = 2, 3$ into a virtual domain $\mathcal{D} \in \mathbb{R}^d$, such that all the boundaries (outer or inner) become circular (2D) or convex (3D), and (2) a tunable greedy routing mechanism to guarantee data delivery between any $s - t$ pairs in $\mathcal{N}$, while providing flexibility to fine tune the tradeoff between energy efficiency and load balancing. In designing these two components of GRIP, we make the following main contributions:

- A fundamentally new parametrization procedure to "regularize" the boundaries (outer or inner) of a multi-connected domain in both 2D and 3D.
- Fully distributed and localized algorithms to perform the geometric parametrization and to obtain virtual coordinates in a WSN.
- A greedy routing objective that combines both the distance and the "position" in the harmonic field resulting from the parametrization, providing tunability towards diversified routing paths.
- A real implementation of GRIP in TOSSIM for feasibility evaluations.

In the following, we first give more detailed discussions on the closely related literature in Sect. 2. We provide an overview of GRIP in Sect. 3, including the network model and the enabling services. Then we focus on GRIP's two main components respectively in Sect. 4 and 5, along with the corresponding analysis. We also discuss related issues in Sect. 6. We report and discuss our simulations results in Sect. 7. Finally, we draw conclusions in Sect. 8.

## 2 Related work

Our discussions in this section only focus on the approaches of applying embedding (or domain mapping) to allow stateless greedy routing. The reason is that, the "local tuning" approaches, such as [1, 8], to enhance the stateless greedy routing is proven to be impossible for 3D routing by Durocher et al. [2]. What is shown in [2] is that routing decision based only on local information may always lead to infinite loop in 3D Euclidean space, which is essentially the consequence of the increased "freedom" in choosing directions in a high dimensional space. Although randomized algorithms may give a temporary cure to this issue [4], it simply cannot provide guaranteed delivery.

One of the earliest approaches of generating virtual coordinates through embedding is proposed in [18]. While the idea there was about location-free routing (rather than getting stateless greedy routing out of local minimums), Rao et al. [18] did motivate the idea of using virtual coordinates to enable stateless greedy routing. The question asked by Papadimitriou and Ratajczak [17] is under what condition(s) a connectivity graph admits a *greedy embedding*, i.e., an embedding in a 2D plane under which stateless greedy routing never fails. Their conjecture was that any planar 3-connected graph admits greedy embedding, and the conjecture was later proven by Leighton and Moitra [11].

One of the important results along the line of research is Kleinberg's statement that every connected finite graph has a greedy embedding in hyperbolic plane [10]. Although this result extends the condition for greedy embedding from 2D to an arbitrarily higher dimension, the idea of embedding network topology graph (a combinatorial structure) has two major drawbacks. Firstly, any change in a network may require a substantial revamp of the embedding; the large overhead makes such a scheme impractical in large scale WSNs [4]: one would be better off using a stateful routing instead. Secondly, as the embedding is performed on a spanning tree, the path between any $s - t$ pair becomes unique and many $s - t$

pairs may share a substantial amount of links with their routing paths. The consequent unbalanced load again makes this scheme unsuitable for the energy constrained WSNs.

Suppose we can *map the space* covered by a WSN to another one with a geometry that favors the greedy routing, changes in network topology would be limited within their neighborhood; they affect the whole map only if certain communication voids that hamper a greedy routing are created. This is the idea brought forward by Sarkar et al. [19], and was later improved with load balancing features by Zeng et al. [22]. Unfortunately, these earlier approaches are confined in 2D spaces, because the geometric tools used there (e.g., *discrete* Ricci flow) are still open for higher dimensions. Even if we consider only 2D WSNs, the heavy computation load imposed by these tools makes themselves less practical. Our approach presented in this paper significantly improves upon these two aspects. Our map works for *both 2D and 3D*. The computations required for sensor nodes are all *arithmetic*; they can be performed in a localized manner and converge quickly. All these features contrast strikingly against the existing approaches that entail computing trigonometric functions and result in a slow convergence rate [19, 22].

## 3 Overview of GRIP

In this section, we first explain the principles that guide the GRIP design. Then we describe the network model and the system construction of GRIP. We also discuss the services required by GRIP to function properly.

### 3.1 Basic principles

We use Fig. 1 to illustrate the principles of GRIP. As shown on the left side, the existence of two lakes in the area covered by a WSN results in two communication voids (of non-convex shapes),[1] causing failure of stateless greedy routing.

To overcome this difficulty, GRIP first parameterizes the original network domain to a virtual domain, in which all the boundaries are convex. Under the coordinates of this virtual domain, we will show that stateless greedy routing always succeeds. However, the contribution of GRIP goes beyond a simple stateless greedy routing, which can lead to a concentration of routing paths at the internal boundaries [22]. As shown in Fig. 1 (the right side), GRIP's tunable greedy routing can get around the holes by having a greedy objective that adapts to the geometry of the network domain.

[1] Geological objects are only one reason for this; irregular deployment of WSNs may be another.

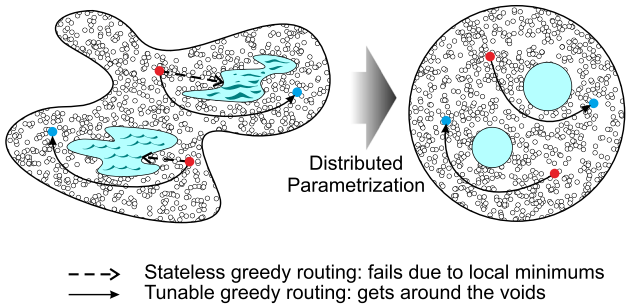


**Fig. 1** A network domain with concave holes (*left*) may lead to routing failure. GRIP maps the domain to a virtual domain (*right*) and applies a greedy routing with tunable objective to achieve load-balancing routing

*Remark* Some of our illustrations/examples are 2D to facilitate visual interpretation. Nevertheless, our simulations will be performed for both 2D and 3D WSN settings.

#### 3.1.1 Network model and system construction

We assume WSN nodes are deployed in 2D or 3D spaces and that two nodes can communicate with each other directly iff their distance is below a certain threshold. Without loss of generality, we set this threshold to one. The resulting graph $\mathcal{G}(V, E)$ is termed *unit disk graph* (UDG) for 2D and *unit ball graph* (UBG) in 3D, where $V$ is the set of nodes and edge $(u, v)$ is in $E$ if $u$ and $v$ can communicate with each other directly. For a 3D WSN, we assume that all its boundaries are genus-0 surfaces. A *stateless greedy routing* relies only on the locations (either exact or virtual) of the source, the destination, as well as every forwarding node $u$ and its one-hop neighbors $N(u)$.

The components of GRIP are as follows:

- Restricted Delaunay Triangulation
- Boundary Detection and Location Service
- Distributed Network Parametrization
- Tunable Greedy Routing

We will briefly introduce the first two components in the following. The remaining components, which are our main contributions in designing GRIP, will be detailed in Sect. 4 and 5, respectively.

#### 3.1.2 Restricted delaunay triangulation

In order to parameterize the network domain to a virtual domain with required properties, the topology of the network domain is needed. GRIP gets aware of this information through the WSN nodes. In other words, using the connectivity relations or the location information (if available), GRIP constructs a Delaunay triangulation upon the network domain, which is a discrete representation of the "shape" of the domain.

Straightforward Delaunay triangulation may result in arbitrarily long edges, hence can fail to represent the topology of a domain. For example, triangular faces with very long edges may eventually cover the holes, hence create an illusion of a simply connected domain. Gao et al. [5] propose a *restricted Delaunay graph* (RDG): it confines its edge set $\tilde{E}$ to be a subset of the network edge set $E$. The outcome is that all the communication voids are represented as non-triangular faces. Although the distributed algorithm proposed in [5] is meant for 2D networks, it can be extended to 3D to construct tetrahedron mesh, again based on the principle of the restricted Delaunay graph.

*Remark* The RDG only serves our distributed parametrization; it is by no means used to assist routing.

#### 3.1.3 Other supporting services

Similar to other routing mechanism relying on virtual coordinates, GRIP needs the support from two services, namely boundary detection and location management. While boundary detection lets a node at a boundary to be aware of its state and the boundary surface it sits on, location management allows any node to query the (virtual) coordinates of other nodes. A location management service is not constrained by the dimensionality, so we can use any existing solutions, such as [15]. An effective 3D boundary detection scheme has recently been proposed [13, 14]. In addition to getting aware of the network boundaries, it also supports distributed coordinations among boundary nodes. For example, a boundary surface can be constructed in a distributed way and certain consensus can be reached among node belonging to the same boundary surface.

## 4 Distributed parametrization in 2D/3D wireless sensor networks

In this section, we focus on the distributed algorithms to realize the map illustrated in Fig. 1. In Sect. 4.1, We first introduce the mathematical background for individual tools, as well as corresponding distributed algorithms to implement these tools. Then we summarize the distributed parametrization procedures for 2D and 3D in Sect. 4.2. We finally present the complexity analysis of the two procedures in Sect. 4.3.

### 4.1 Mathematical background and algorithms

We hereby explain the individual mathematical tools, namely harmonic function, gradient vector field, integral curve, and annulus embedding. For each tool, we propose a distributed algorithm to perform the computation; these algorithms are executed by individual nodes and they require only the access of local information. To simplify the exposition, we first introduce the concepts and tools for 2D cases in Sect. 4.1.1 to 4.1.4, and then discuss the generalization to $\mathbb{R}^3$ in Sect. 4.1.5.

#### 4.1.1 Harmonic function

Given a network domain $\mathcal{N} \subseteq \mathbb{R}^2$ with holes, we consider it as a multi-connected domain with boundaries $\partial\mathcal{N} = \gamma_0 - \gamma_1 - \cdots - \gamma_l$, where $\gamma_0$ is the outer boundary and $\gamma_i$, $1 \le i \le l$, are the inner hole boundaries. These outer and inner boundaries can be identified in a localized manner [14]. The harmonic function $f : \mathcal{N} \to \mathbb{R}$ is twice continuously differentiable and satisfies the Laplace's equation $\triangle f(p) = 0$, $\forall p \notin \partial\mathcal{N}$, with Dirichlet boundary condition $f|_{\gamma_i} = f_i$, $0 \le i \le l$.

According to Sect. 3.1.2, $\mathcal{N}$ is represented by an a triangle mesh (i.e., a RDG) $M = (V, \tilde{E}, F)$ where $V$, $\tilde{E}$ and $F$ are the set of vertices, edges and faces, respectively. As pointed out in [19], the boundaries $\partial M$ of the mesh may well approximate $\partial\mathcal{N}$. Let $\tilde{E}(p)$ be the subset of $\tilde{E}$ incident to $p$, the discrete Laplace operator of an interior vertex $p \notin \partial M$ is defined as $\triangle f(p) = f(p) - \sum_{(p,q)\in\tilde{E}(p)} \omega_{pq} f(q)$, where $\omega_{pq}$ is the weight of directed edge $(p, q)$. The choice of the weight is usually application dependent. We choose a straightforward weight determined by vertex degree, i.e., $\omega_{pq} = \frac{1}{deg(p)}$, with $deg(p)$ being the degree of vertex $p$.

Observing the above Laplace equation is equivalent to the heat diffusion

$$\frac{\partial f}{\partial t} = -\triangle f, \tag{1}$$

we can solve it using the Euler method in a distributed manner, as shown by Algorithm 1. Given the boundary conditions $f_0 = 0$ and $f_i = 1, i \neq 0$, The initial and final states of a domain are shown in Fig. 2(a) and (b).

**Algorithm 1:** Computing harmonic function

**Input**: The Dirichlet boundary condition $f_i$. For each $p \in V$, $\tilde{E}(p)$, the step length $\delta$, the stopping tolerance $\varepsilon$
**Output**: A harmonic function $f : M \to \mathbb{R}$

```
Initialize f(p) ← f_i, p ∈ γ_i; otherwise f(p) ← 0
For every interior vertex p periodically:
if |△f(p)| > ε then
    △f(p) = f(p) − Σ_{(p,q)∈Ẽ(p)} ω_pq f(q)
    f(p) ← f(p) − δ △f(p)
    Broadcast f(p) to all q : (p, q) ∈ Ẽ(p)
end
```

#### 4.1.2 Gradient vector field

Given an arbitrary smooth function $g$ defined on $\mathcal{N}$, we can approximate the gradient $\nabla g$ using $M$. For a triangle $\triangle p_1 p_2 p_3$, let $g_i = g(p_i)$, $1 \le i \le 3$, be the function value

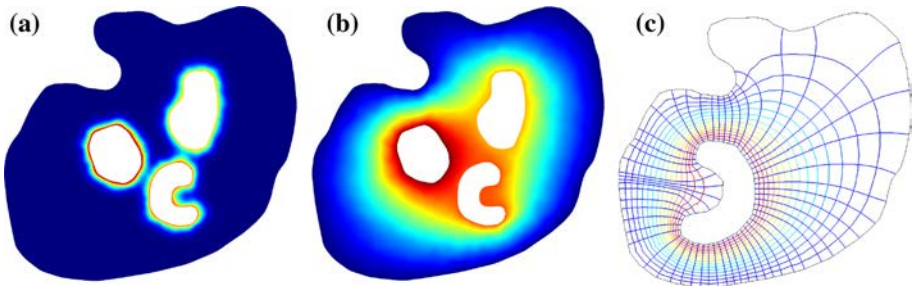


**Fig. 2** Harmonic function and integral curves. **a**, **b** Computing the harmonic function for a WSN with three holes. **c** The harmonic function defined on a topological annulus has no singularity and the integral curves of the harmonic function are perpendicular to the iso-value curves. **a** Initial status,**b** convergence, **c** Integral curves

defined for each vertex. Let $\overrightarrow{p_ip_{i+1}}$ be the vector from $p_i$ to $p_{i+1}$ (the modulo-3 operator is applied to the subscript index). The gradient $\nabla g$ inside $\triangle p_1p_2p_3$ is

$$\nabla g|_{\triangle p_1p_2p_3} = \sum_{i=1}^{3} r(\overrightarrow{p_{i+1}p_{i+2}})g(p_i),$$

where $r(\overrightarrow{v})$ is the counter-clockwise rotation of $\overrightarrow{v}$ in 90°.

Given a vertex $p$, let $Nb(p)$ denote the triangles adjacent to $p$. We first compute the gradient for each triangle in $Nb(p)$ and then compute the gradient on $p$ by weighted sum:

$$\nabla g(p) = \frac{\sum_{t\in Nb(p)} s(t)\nabla g|_t}{\sum_{t\in Nb(p)} s(t)},$$

where $s(t)$ measures the area of triangle $t$.

#### 4.1.3 Tracing integral curves

Suppose $\mathbf{F}$ is a 2D vector field defined on $M$, i.e., $\mathbf{F} : M \to \mathbb{R}^2$, and $\mathbf{x}(\tau)$ a parametric curve. Then $\mathbf{x}(\tau)$ is an integral curve of $\mathbf{F}$ if it satisfies $\mathbf{x}'(\tau) = \mathbf{F}(\mathbf{x}(\tau))$. Intuitively speaking, the integral curve is a curve such that the tangent vector to the curve at any point $p$ along the curve is precisely the vector $\mathbf{F}(p)$. Given a triangle $\triangle p_1p_2p_3$ and an arbitrary point $q \in \triangle p_1p_2p_3$, we can compute the integral curve in a localized manner as shown in Algorithm 2, where $Barycentric(q|\triangle p_1p_2p_3)$ returns the barycentric coordinates of $q$ with respect to $t$. Note that only one of the vertices of $t$ needs to perform the computation, which involves purely arithmetic operations. The collective outcome of tracing a gradient vector field (i.e., integral curves and iso-value curves) in a WSN is shown in Fig. 2(c).

**Algorithm 2:** Tracing inside a triangle

**Input**: A triangle $t = \triangle p_1p_2p_3$, the vector $\mathbf{F}(p_i)$, $i = 1, 2, 3$, a point $q \in t$, the step length $\delta$
**Output**: An integral curve $\gamma \in t$ following the vector field $\mathbf{F}$, formed by the locus of $q$

```
while q ∈ t do
    (λ1, λ2, λ3) = Barycentric(q| △ p1p2p3)
    F(q) = Σ_{i=1}^{3} λ_i F(p_i)
    q ← q + δ * F(q)
end
```

#### 4.1.4 Annulus embedding of 2-connected domain

We have the following proposition of a harmonic function on the 2-connected domain:

**Proposition 1** *Let $M$ be a 2-connected domain $\partial M = \gamma_0 \bigcup \gamma_1$ and $f : M \to \mathbb{R}$ be a harmonic function with boundary conditions $f|_{\gamma_0} = 0$ and $f|_{\gamma_1} = 1$. The gradient vector field $\mathbf{F} = \nabla f$ has the following properties: 1) any integration curve $\mathbf{x}$ is an open curve with two ending points on $\gamma_0$ and $\gamma_1$ respectively; 2) any two integration curves $\mathbf{x}_1$ and $\mathbf{x}_2$ do not intersect; and 3) for any point $p \in M$, there is a unique integration curve $\mathbf{x}$ passing through $p$.*

The proof is presented in the Appendix 1. According to these properties, we can naturally embed the 2-connected domain to an annulus in a conformal (i.e., angle-preserving) manner. We first calculate all integral curves from outer boundary to the inner boundary, then we map the inner boundary to an inner circle of annulus. Finally we map each integral curve to corresponding radius on annulus. For the discrete case (where a triangular mesh $M$ is concerned), we only need to calculate the integral curve from each vertex to the inner boundary. We can determine the vertex location on the annulus by polar coordinates $(\rho, \theta)$, where $\rho$ is determined by the harmonic function value $f$ and $\theta$ is determined by the end point of integral curve on the inner boundary.

Since for each vertex, $f$ is known and we only need to compute $\theta$ in polar coordinates, the embedding can be accomplished in a distributed and efficient way, by tracing part of the integral curve (rather than the whole curve from vertex to inner boundary). Basically, a tracing can be stopped when it meets an edge whose $f$ at two endpoints are both larger than the $f$ of the starting vertex. Then the stopping position on the edge is recorded, and the $\theta$ of this position (which is in turn the $\theta$ of the starting vertex) is obtained by linear interpolation. As a vertex with lower $f$ always obtain its $\theta$ from some vertices that have bigger $f$, a distributed iteration may start from the vertices on the inner boundary, whose $f$ is highest, and gradually propagate to the outer boundary. We can always obtain the $\theta$ for all

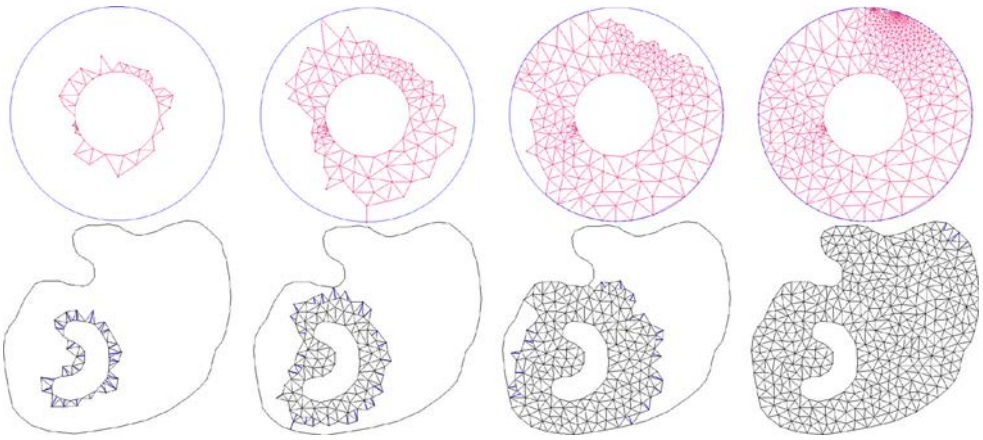

**Fig. 3** Propagative embedding a 2-connected triangle mesh into an annulus

vertices after the iteration reaches the outer boundary. This distributed embedding procedure is illustrated in Fig. 3 and is shown by Algorithm 3, where $l(p,q)$ (line 14) returns the length of edge $(p,q)$.

**Algorithm 3:** Annulus embedding

**Input**: A 2-connected triangle mesh $M \in \mathbb{R}^2$ with two boundaries $\partial M = \gamma_0 \bigcup \gamma_1$
**Output**: A conformal map $\phi$ that maps $M$ to an annulus, i.e., $\phi(p) = (\rho_p, \theta_p)$, where $(\rho_p, \theta_p)$ is the polar coordinate.

1 Compute a harmonic function $f : M \to \mathbb{R}$ with Dirichlet boundary condition $f|_{\gamma_i} = i$, $i = 0, 1$, using Algorithm 1
2 Choose a vertex $p_0 \in \gamma_1$ and set its $\theta_{p_0} = 0$
3 Compute $\ell = \int_{\gamma_1} dl$, the length of $\gamma_1$
4 **if** *vertex* $p \in \gamma_1$ **then**
5 $\quad l_p$ = length from $p$ to $p_0$ on $\gamma_1$ in counter clockwise direction; $\theta_p = \frac{l_p}{l}$
6 **else**
7 $\quad$ Apply Algorithm 2 to trace the integral curve from $p$ to an edge $(a,b)$, s.t. $f(a) > f(p)$ and $f(b) > f(p)$
8 $\quad$ Record the stopping position $s_p$
9 $\quad$ Initialize the $\theta$ of $p$ to $NaN$
10 **end**
11 For every interior vertex $p$:
12 Periodically check all $(p,q) \in \tilde{E}(p)$ for which at least one stopping position $s_{\bar{p}}$ is recorded
13 **if** $\theta_{\bar{p}} = NaN$, $\theta_p \neq NaN$, $\theta_q \neq NaN$ **then**
14 $\quad \theta_{\bar{p}} = \theta_p \frac{l(p,s_{\bar{p}})}{l(p,q)} + \theta_b \frac{l(s_{\bar{p}},q)}{l(p,q)}$; Update vertex $\bar{p}$ with $\theta_{\bar{p}}$
15 **end**
16 **return** $(\rho_p = f(p), \theta_p)$

The first part of the algorithm (lines 2 to 5) needs distributed coordinations along the inner boundary, we refer to Sect. 3.1.3 for details. The second part involves curve tracing (lines 6 to 10) and $\theta$ value propagation (lines 11 to 15). It can be shown that none of them require transmissions beyond a single hop, so the algorithm is by default localized.

#### *4.1.5 Extension to 3D*

We have shown that computing discrete harmonic function and its gradient on triangular mesh embedded in $\mathbb{R}^2$. This machinery can be easily generalized to a tetrahedral mesh in $\mathbb{R}^3$. Let $M = (V, \tilde{E}, F, T)$ be a tetrahedral mesh, where $T$ is the set of tetrahedral. Computing harmonic function on tetrahedral mesh can also be done by Algorithm 1; the outcome is shown by Fig. 4(a)–(c).

Computing the gradient in tetrahedral mesh is similar to the 2D counterpart. Given a tetrahedron $p_1p_2p_3p_4$, let $\overrightarrow{v}_i$, $1 \leq i \leq 4$, be a vector, which is perpendicular to triangle $\triangle p_{i+1}p_{i+2}p_{i+3}$ and points outward of the tetrahedron. The magnitude $\|\overrightarrow{v}_i\|$ equals the area of triangle $\triangle p_{i+1}p_{i+2}p_{i+3}$. Then, the gradient $\nabla g$ inside tetrahedron $p_1p_2p_3p_4$ is $\nabla g|_{p_1p_2p_3p_4} = \sum_{i=1}^{4} \overrightarrow{v}_i g(p_i)$. Computing the gradient for each vertex is similar to the 2D case except that tetrahedron volume is used in the weighted sum.

When embedding a 2-connected domain in 2D to an annulus, we need to map the inner boundary to a circle (lines 2-5 in Algorithm 3). Let $D_r = \{x \in \mathbb{R}^3 | r \leq \|x\|_2 \leq 1\}$ be the 3D embedding domain. We need to map the inner boundary (surface) $\gamma_1$ to a sphere with radius $r$. This can be tackled by using Algorithm 1 to solve a harmonic function, so it is again localized and involves only arithmetic operations. Details are omitted due to limited space. We also need to compute another harmonic function $f : M \to \mathbb{R}$ with Dirichlet boundary condition $f|_{\gamma_0} = 0$ and $f|_{\gamma_1} = 1$. Then the gradient $\nabla f$ has exactly the same properties as its 2D counterpart (see *Proposition* 1). Therefore, similar to Algorithm 3, we can embed $M$ to $D_r$ by tracing the integral curves, as shown in Fig. 4(d) and (e).

### 4.2 Distributed parametrization

With the set of algorithms presented in Sect. 4.1, we can map a network domain with one hole to an annulus. We now present Algorithm 4 that parameterizes a network domain with many holes, by iterating the single-hole map (Algorithm 3) among all the holes. We also illustrate the algorithm in Fig. 5 and the outcome for a 3D domain in Fig. 6.

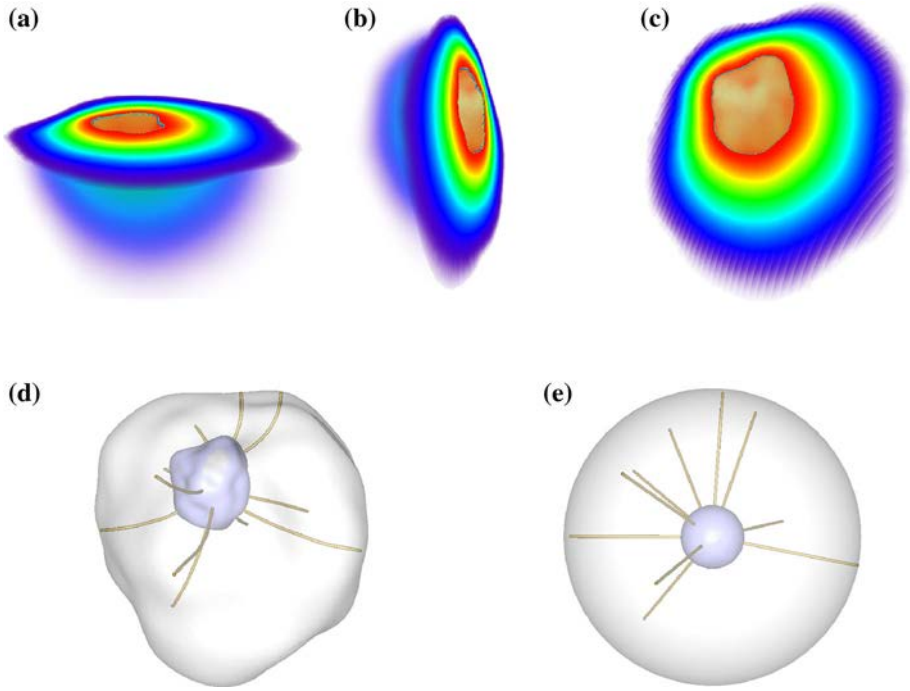


**Fig. 4** The cut views of the harmonic function on the 3D network domain with a void (**a**)–(**c**). Integral curves in a network domain (**d**) and in the embedding domain, i.e., a solid ball with a spherical hole in the center (**e**). **a** x-cut view, **b** y-cut view, **c** z-cut view, **d** network domain, **e** embedding domain

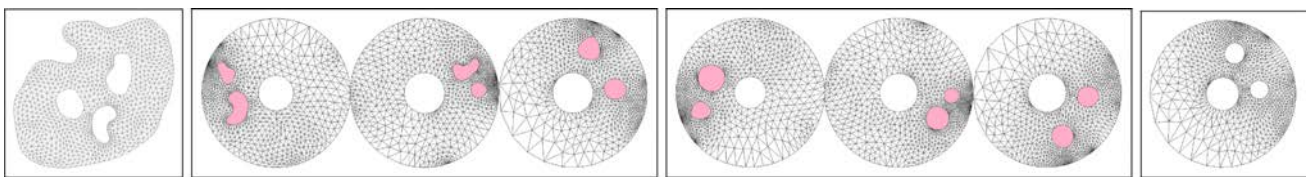

**Fig. 5** Distributed parametrization of a 2D WSN. Each iteration of the algorithm consists of three steps, since the given 2D WSN contains three holes. Initially, all holes are filled virtually. In each step, we open one hole to get a 2-connected domain, and then apply Algorithm 3 to parameterize it to an annulus. The original network domain, the intermediate results, and the final parametrization after two iterations are shown from *left* to *right*

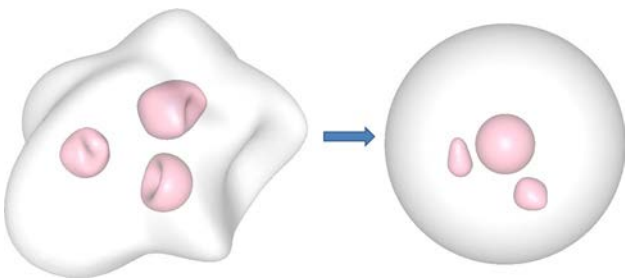

**Fig. 6** Parametrization of a 3D WSN where one void is mapped to a sphere centered at the origin and others are mapped to convex polyhedra

**Algorithm 4:** Iterative parametrization of a domain with multiply genus-0 boundaries

**Input**: A domain $M$ with multiply boundaries
**Output**: A bijective parametrization $\phi$ mapping $M$ such that all boundaries become convex

1 Fill all holes virtually
2 **while** *Some hole boundary* $\gamma_i$ *is not convex* **do**
3 | Open the $i$-th hole virtually
4 | Perform Algorithm 3 in terms of $\gamma_i$
5 **end**

The procedures of filling and opening holes virtually (lines 1 and 3) simply require an inner boundary vertex to identify a corresponding vertex on the the boundary and to bridge, with virtual edges, all the vertex pairs or to cut the

virtual edges. These can be achieved with the support of a boundary detection service [14]. The convexity test in line 2 can be performed in a localized manner. In particular, for each vertex $p \in \gamma_1$, its neighborhood $N(p)$ in $\mathcal{G}$ should contain a subset $\bar{N}(p) \subset \gamma_1$. Let $\alpha_p$ be the angle (or solid angle in 3D) determined by $\{p\} \cup \bar{N}(p)$, then the convexity test returns true of $\alpha_p < \pi$ (or $\alpha_p < 2\pi$ in 3D). In general, we have

**Proposition 2** *Our iterative parametrization algorithm terminates with all boundaries being convex.*

The proof is sketched in Appendix 2. In fact, for 2D WSNs, we can use circularity as the termination condition, such that the algorithm terminates with all boundaries being circular.

### 4.3 Message complexity analysis

Given a 2D/3D WSN with $l$ holes, Algorithm 4 parameterizes each hole to a circular domain by virtually filling the other holes. It repeats this process until all holes become convex. Let $N_c$ denote the number of times Algorithm 3 is applied to a hole. We have $N_c = \mathcal{O}(1)$. As shown in Fig. 5, only two iterations are required to map each hole to a circular domain.

Note that Algorithm 3 contains two key steps, i.e., computing the harmonic function by using Algorithm 1, and embedding the WSN to an annulus (applying Algorithm 2 to each triangle). The complexity of Algorithm 1 is $\mathcal{O}(\log n)$, where $n$ is the number of nodes in a WSN, given the diffusion nature of the algorithm [9]. The complexity of embedding depends on the network diameter, as the embedding is "propagated" from some inner boundary to the outer boundary. Therefore, we have $\mathcal{O}(\sqrt{n})$ for 2D and $\mathcal{O}(n^{\frac{1}{3}})$ for 3D. Putting them all together, the complexity of our algorithm is $\mathcal{O}(l(\log n + \sqrt{n}))$ for 2D and $\mathcal{O}\left(l\left(\log n + n^{\frac{1}{3}}\right)\right)$ for 3D.

## 5 Tunable greedy routing

With the parametrization procedure applied to a WSN, we are now ready to present our routing protocol under the resulting virtual coordinates. In this section, we first prove, in Sect. 5.1, that stateless greedy routing never fails under the virtual coordinates computed through the distributed parametrization. Then we present an augmented greedy routing in Sect. 5.2: it has a tunable greedy objective that leverages on the harmonic field computed during the parametrization phase (see Sect. 4.1.1), and can hence avoid concentrating the routing load on the hole boundaries.

### 5.1 Guaranteed delivery

A stateless greedy routing, as defined in Sect. 3.1.1 needs the locations of the destination $d$, the current forwarding node $u$, and its one-hop neighbors $N(u)$. The greedy forwarding decision at each hop $u$ in general takes the following form:

$$\hat{v} = \arg\min_{v \in N(u)} dist(v, d), \tag{2}$$

where $dist(v, d)$ returns the Euclidian distance between $v$ and $d$, and $\hat{v}$ indicates the next hop. Here the *greedy objective* is $dist(v, d)$ and the *decision oracle* returns $\hat{v}$ that minimizes the objective. Due to the existence of holes in a WSN, oracle (2) may return $\{u\}$ and thus lead to a routing failure. To cope with this issue, we add another criterion in case of failure:

$$\hat{v} = \arg\min_{v \in N(u)} (\mathbf{d} - \mathbf{v}) \cdot \mathbf{d}, \tag{3}$$

where the vector is defined with the source node $s$ as the origin. As we have shown in Sect. 4.2, GRIP's distributed parametrization results in virtual coordinates under which all the boundaries are convex. Now we show that stateless greedy routing never fails.

**Proposition 3** *If a network, under certain virtual coordinates, has all its boundaries convex, our stateless greedy routing always guarantees data delivery.*

*Proof* The correctness is obvious if oracle (2) never fails, so we only need to show how (3) may overcome the situation where (2) fails. The only possibility that (2) fails is that the local angle (or solid angle for 3D) becomes close to $\pi$ (or $2\pi$ for 3D) [3]. However, this also indicates that the routing path has hit a boundary [5], while a boundary is now convex according to *Proposition* 2. The convexity further suggests that, for any node $u$ on the boundary, there exists $v \in N(u)$ such that $(\mathbf{d} - \mathbf{v}) \cdot \mathbf{d} < (\mathbf{d} - \mathbf{u}) \cdot \mathbf{d}$. Therefore, the greedy objective (3) always drives the routing path to make progress along $\mathbf{d}$, hence away from the starting point $u_1$ (see Fig. 7).

Consequently, no routing loop can be produced by (3). As soon as (3) brings the routing path sufficiently far from the node where (2) fails (e.g., to node $v$ in Fig. 7), oracle (2) will return successfully and hence the protocol will switch back to (2). The existence of such a "switching" node can be proven by contradiction. Assume it does not exist, then one can trivially show that $d$ is in the hole, a contradiction.

For each $u \in V$ not on a boundary, there always exists a $v \in N(u)$ such that $d(v, d) < d(u, d)$ under the virtual coordinate, therefore,

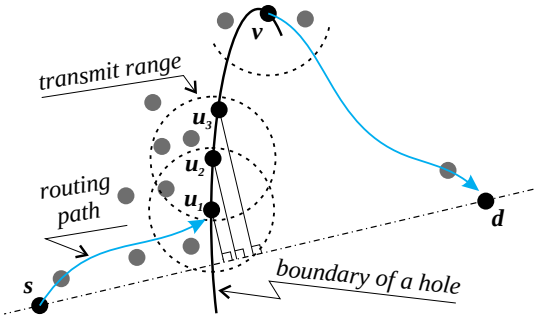


**Fig. 7** Greedy forward decision (3) always succeeds and is loop-free on a convex boundary

**Corollary 1** *The decision oracle in (2) only needs to search within* $N^+(u) \subset N(u)$, *such that* $d(v,d) < d(u,d), \forall v \in N^+(u)$.

### 5.2 Tuning greedy objective

Although stateless greedy routing never fails under GRIP's virtual coordinates, many routing paths could be concentrated along the inner boundaries. This unfortunate phenomenon is a direct consequence of the decision oracle (2): as distance is the only greedy objective, many routing paths are forced to turn away from the line segment between source and destination only upon "hitting" a hole boundary.

Our solution is to slightly change the greedy objective for non-boundary nodes by taking into another "coordinate" into account. One byproduct of the distributed parametrization is a scalar field[2] given by the harmonic function $f(\cdot)$. Specifically, every node $u \in V$ has a scalar value $f(u) \in [0,1]$ that indicates its "distance" towards some inner boundary: the larger the $f(u)$, the closer $u$ is to that inner boundary. We define a new decision oracle by using $f(u)$:

$$\hat{v} = \arg\min_{v \in N^+(u)} (1 + \lambda |\Delta f_{vd}|) dist(v,d), \tag{4}$$

where $\Delta f_{vd} = f(v) - f(d)$. As $\hat{v} \in N^+(u)$, the distance toward $d$ keeps reducing and thus the correctness of this routing algorithm is guaranteed. The rationale behind the greedy objective $(1 + \lambda |\Delta f_{v,d}|) dist(v,d)$ is that, as a routing path whose harmonic values deviate from $f(d)$ will be penalized, routing paths tend to be bent by *isosurfaces* rather than by inner boundaries. Consequently, the routing paths are evenly distributed instead of being concentrated at the inner boundaries. By tuning the weight $\lambda$, we may choose a proper tradeoff between energy efficiency and load balancing: a larger value of $\lambda$ biases towards tracing the isosurfaces rather than the line from source to destination, resulting in better load balancing but longer routing paths.

In order to better adapt to the geometry of the network domain, we may tune $\lambda$ online, as shown by Algorithm 5.

**Algorithm 5:** Online Greedy Objective Tuning

```
v̂ = arg min_{v∈N⁺(u)} (1 + λ_u|Δf_vd|) dist(v,d)
if Δf_v̂u = f(v̂) − f(u) > 0 then
    λ_v̂ = incr(λ_u)
else
    λ_v̂ = decr(λ_u)
end
Piggyback λ_v̂ with the packet to be forwarded to v̂
```

Basically, the $\lambda_{\hat{v}}$ for $\hat{v}$ (the next hop) is tuned by the current forwarding node $u$ according to $\Delta f_{\hat{v}u}$. On one hand, $\lambda$ is increased to penalize a positive $\Delta f_{\hat{v}u}$, as the routing path tends to approach some inner boundary. On the other hand, $\lambda$ is decreased to encourage a negative $\Delta f_{\hat{v}u}$, as it either helps to improve load balancing or allows the routing path to "jump" from one isosurface to another; this later jumping effect is necessary to maintain a reasonable routing stretch when the isosurface of the source is not connected (in topological sense) to that of the destination.

## 6 Discussions

As GRIP's virtual coordinates are obtained by embedding the network domain, they are very insensitive to changes in network topology. Nodes may join or leave the network, or they may move within the network domain. As far as no new holes are created, such changes in network topology will not cause another distributed parametrization procedure. Nodes that are affected by such changes only need to locally reconstruct the RDG and interpolate their virtual coordinates from their neighbors. As demonstrated by Fig. 8, the cost to maintain the virtual coordinates is often negligible. On the contrary, the approach that embeds the network topology [10], though appears to be lightweight during the parametrization phase, is very sensitive to changes in the topology: virtually any slight changes can lead to the revamp of the embedding.

Though discrete surface Ricci flow [19] is an effective technique to parameterize multiply connected surfaces by manipulating the target curvature, GRIP is superior to [19] in two aspects: First, the distributed version of the 2D discrete Ricci flow method is computationally expensive for sensor nodes, as trigonometric functions and square root are involved. GRIP, in sharp contrast, only entails simple arithmetic functions, and it is thus much more efficient. Second, although the continuous version of Ricci flow for 3-manifolds has been widely studied, defining the discrete Ricci flow on tetrahedral meshes is still an open

[2] The vector field will be discarded after the parametrization, as it is used to assist the parametrization and is of no use to a greedy routing.

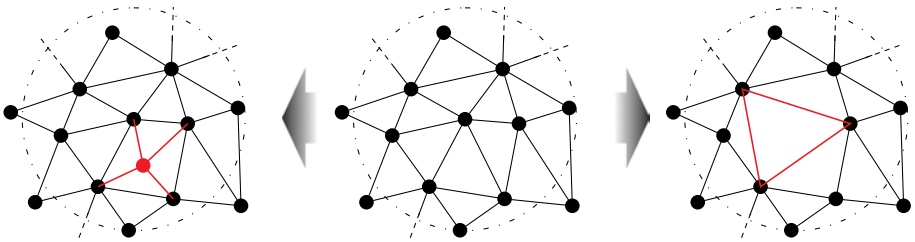

**Fig. 8** Changes in network topology due to node joining (*left*) and leaving (*right*) only have local influences. The *dashed circle* represents the transmission range of a node

problem. GRIP is based on tracing integral curves, which is well defined in $\mathbb{R}^n$. We will compare their complexity in Sect. 7.1. One recent proposal directly applies a volumetric harmonic map to 3D WSNs [21]. However, it can handle only WSNs with at most one hole, whereas our proposal is much more general as it works for any number of holes.

Our approach also fundamentally differs from Koebe's method [23], in that Koebe's method computes a holomorphic 1-form (a complex 1-form such that both real and imaginary parts are harmonic) on the 2-dimensional surface. However, it is known that a 3D volume does not have holomorphic 1-form. Thus, the holomorphic 1-form based Koebe's method cannot be generalized to parameterize 3D volumes. Moreover, computing holomorphic 1-form requires solving a sparse linear system, whose distributed implementation can be hard.

One potential drawback of our GRIP framework is that its 3D part has not been able to take into account the irregularities in network connectivity. In 2D, such irregularities are captured by extending UDG to Quasi UDG, and parametrization algorithms relying on triangulation can be adapted to Quasi UDG [19]. However, there is no extension to 3D UBG yet, and it is an open problem that what impact could be brought to triangulation by such an extension.

## 7 Simulations and experiments

In this section, we evaluate the performance of GRIP through simulations. We have implemented GRIP in TOSSIM [12]. However, as TOSSIM has difficulty in simulating large scale WSNs (e.g., tens of thousands of nodes), we have also developed a high level simulator (which neglects the MAC effects) for those WSNs.

We evaluate the distributed parametrization phase and the routing phase separately. We first compare our parametrization with the Ricci flow approach [19] in Sect. 7.1. Then we evaluate the performance of different greedy routing schemes under the virtual coordinates in Sect. 7.2.

### 7.1 Parametrization efficiency

We compare GRIP's parametrization with the distributed Ricci flow method [19] in terms of time complexity. As

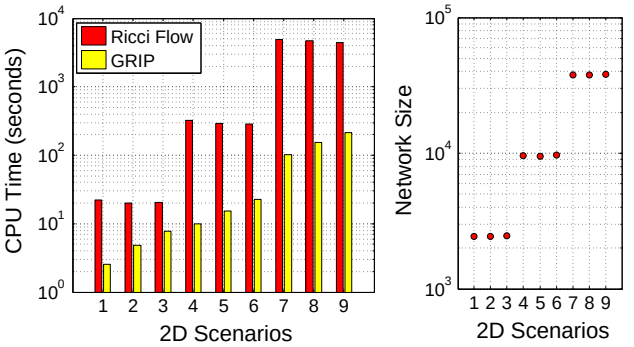


**Fig. 9** CPU time comparison between GRIP's parametrization and Ricci flow. $l_1 = l_4 = l_7 = 2$, $l_2 = l_5 = l_8 = 3$, and $l_3 = l_6 = l_9 = 4$, where $l_i$ is the number of holes in the $i$-th 2D WSN

**Table 1** Simulated WSNs

| S. no | No of nodes | No of holes | S. no | No of nodes | No of holes |
|---|---|---|---|---|---|
| 1 | 9299 | 3 | 2 | 1089 | 2 |
| 3 | 10433 | 3 | 4 | 9654 | 4 |
| 5 | 8358 | 2 | 6 | 9146 | 4 |
| 7 | 8750 | 2 | 8 | 2628 | 2 |
| 9 | 79999 | 1 | 10 | 80000 | 2 |
| 11 | 80000 | 3 | 12 | 80000 | 4 |

discussed in Sect. 6, the latter only works for 2D meshes for now, so the comparison is done in nine 2D WSNs with different sizes. As the Ricci flow method has not been implemented under the TinyOS framework, we only compare these two algorithms in a simulated environment. This performance comparison is shown in Fig. 9. Ricci flow differs from GRIP's parametrization in that it is able to deal with all holes in a WSN simultaneously. However, as Ricci flow involves trigonometric function and square root function, and it usually requires many iterations to terminate, the overall performance is much worse.

### 7.2 Routing performance

We evaluate the greedy routing performance in this section. The three routing protocols concerned in this paper are:

1. Pure Greedy Routing: with decision oracle (2). This is also the method used in [19]

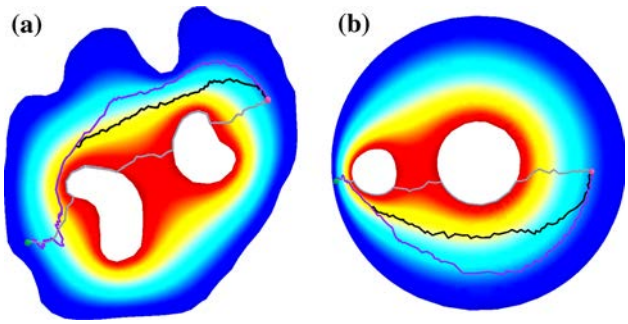


**Fig. 10** Comparing different greedy routing schemes. The 2D WSN (*left*) is mapped to the parametric domain (*right*). While the pure greedy scheme (*gray curve*) sticks to the inner boundary, the two tunable schemes (*purple* and *black curves*) obviously tend to follow the isosurfaces, and the online tunable scheme (*black curve*) appears to achieve the most balanced performance. **a** Network domain, **b** Embedding domain (Color figure online)

2. Offline Tunable Greedy Routing: with decision oracle (4) and $\lambda$ is set offline.
3. Online Tunable Greedy Routing: with decision oracle (4) and $\lambda$ is tuned online using Algorithm 5.

As proven in Sect. 5.1, all these routing schemes always succeed, and this is also what we observed in our experiments. Therefore, we compare them with respect to other metrics, namely, *load balancing*, *routing path stretch*, and *total energy consumption*. We assume that, as soon as a routing path goes through a node, the load (defined as energy consumption) taken by this node is increased by one unit. Therefore, given a certain (large) number of routing paths, the load balancing is verified by checking the maximum load taken by a node, whereas the total energy consumption is the sum of the load for all nodes.

The features of the WSNs we worked on are listed in Table 1. For each WSN, the sensor nodes are uniformly distributed in the deployment field. The first 8 cases are 2D WSNs, while the remaining are 3D WSNs. Before comparing these WSNs against the three metrics, we first use Net5 to illustrate the behavior of the three routing schemes in Fig. 10. The three routing paths between two arbitrary nodes are drawn in both the network domain and embedding domain, with the harmonic field as the background. Though they all succeed while facing non-convex holes, pure greedy scheme is "attracted" by the inner boundaries, while the online tunable scheme makes the best tradeoff between energy efficiency and load balancing.

The load balancing effects of the three routing schemes are shown in Fig. 11(a), for both 2D (upper) and 3D (lower) WSNs. The straightforward observation is that, while the two tunable routing schemes have rather similar performance, the maximum load of the pure greedy routing

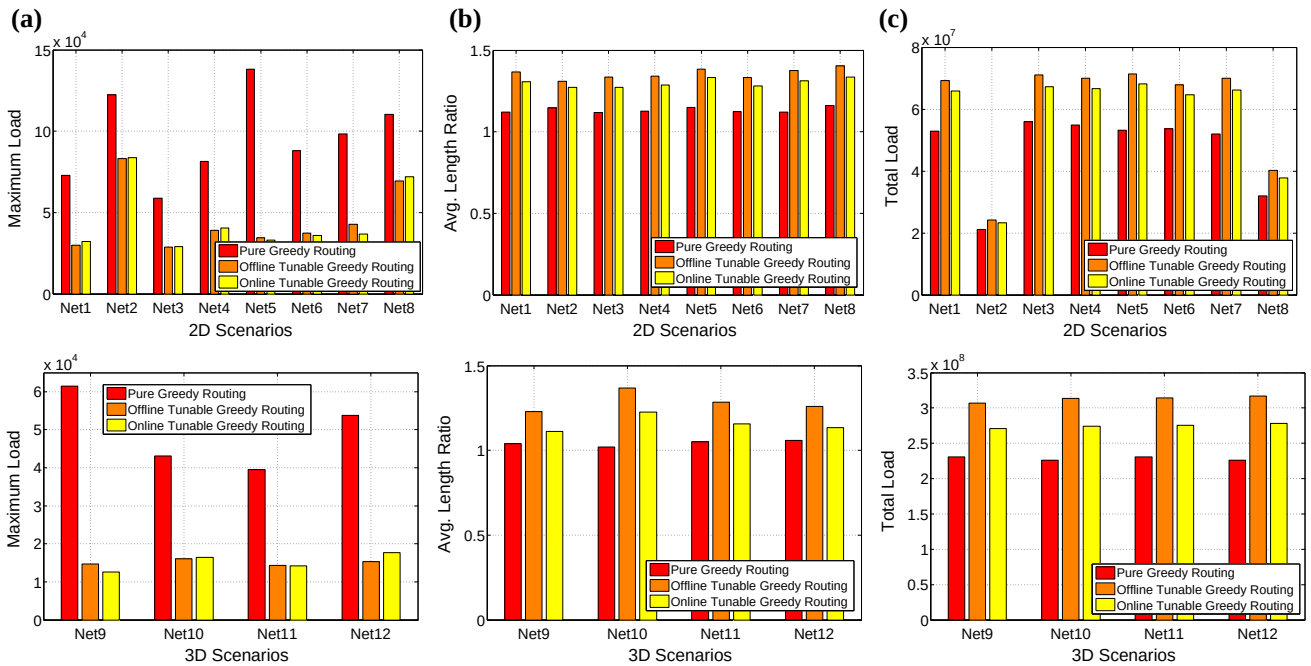


**Fig. 11** Comparing three greedy routing schemes in 2D WSNs (*top*) and 3D WSNs (*bottom*). **a** Load balancing. **b** Routing path stretch. **c** Total energy consumption

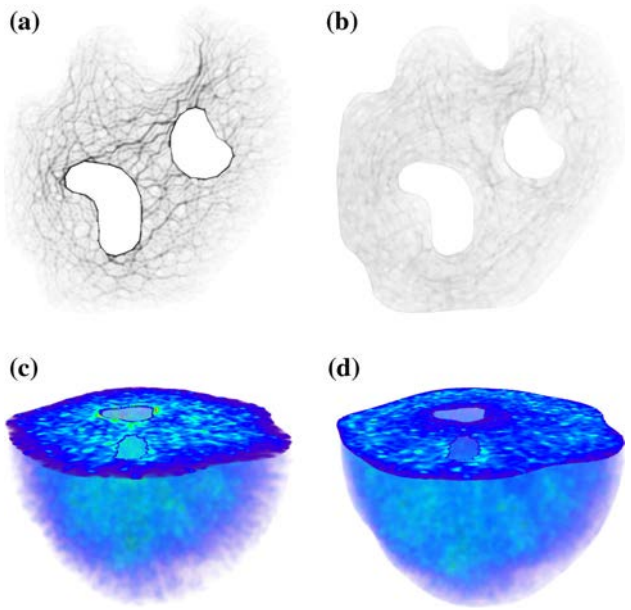


**Fig. 12** Illustration of the load balancing effect in two WSNs. Cut views are applied to reveal the loads inside the 3D WSN. **a** Pure Greedy (Net5). **b** Online Tunable (Net5). **c** Pure Greedy (Net10). **d** Online Tunable (Net10)

is often much higher (it can be up to 3.8 times of those of the other two schemes). We also compare the load balancing effects of the pure greedy and the online tunable schemes in Fig. 12. Obviously, in both 2D and 3D WSNs, the online tunable scheme achieves almost perfect load balancing, whereas the pure greedy scheme generates many hot spots all over the network domain, in particular around the inner boundaries.

Of course, the load balancing effects of the two tunable schemes come at a cost: longer routing paths and hence higher total energy consumption for a whole WSN. However, as shown in Fig. 11(b) and (c), such a cost is worthy of trading for the significant improvement in load balancing. Especially for the online tunable scheme, only up to 30% of the cost is sacrificed in path length or in total energy consumption.

## 8 Conclusion

We presented GRIP (Greedy Routing through dIstributed Parametrization), a greedy routing algorithm with guaranteed delivery in 2D/3D WSNs. Based on a novel distributed parametrization algorithm, we can embed a complicated 2D/3D network domain to a simple parametric domain, in which each hole is mapped to a circle or convex domain. This property favors greedy routing and allows it to have guaranteed delivery. We further proposed two tunable greedy routing schemes to combat the load concentration caused by the pure greedy routing that sticks to the distance greedy principle. We proved the correctness and efficiency of GRIP, and demonstrated that our tunable greedy routing schemes are able to naturally generate routing path diversity. Using simulations in TOSSIM, we demonstrated the tunable greedy schemes can significantly improve load balancing while only marginally sacrificing the energy efficiency.

## Appendix 1: Proof of Proposition 1

Note $f$ is smooth function, its gradient vector fields are curl-free. Thus, no integral curve can form a loop inside $M$. Furthermore, the function $f$ is harmonic and there is no critical points (where the gradient vanishes) inside $M$. Thus, the function value is strictly monotonic along the integral curve. Note that all points on the same boundary curve have the same function value, so the ending points of each integral curve must be on different boundary curve.

Then we show that two integral curves do not intersect. Assume two integral curves $\mathbf{x}_1 \in M$ and $\mathbf{x}_2 \in M$ intersect at a point $p$. Then $p$ is a critical point and the gradient $\nabla f$ vanishes at $p$. Since $f$ is harmonic, the maximum and minimum must be on the boundaries. Therefore the Hessian matrix at $p$ has negative eigenvalue values. Suppose $f(p) = s$, then according to Morse theory, the homotopy types of the level sets $f^{-1}(s-\epsilon)$ and $f^{-1}(s+\epsilon)$ will be different, where $\epsilon$ is a small positive value. At all the interior critical points, the Hessian matrices have negative eigenvalues, the homotopy type of the level sets will be changed. Therefore, the homotopy type of $\gamma_0$ is different from that of $\gamma_1$. This contradicts the given condition that $M$ is 2-connected. Therefore $\mathbf{x}_1$ and $\mathbf{x}_2$ have no intersection points anywhere.

## Appendix 2: Correctness of Algorithm 4

Due to the page limit, we only sketch the proofs.

**2D WSN.** Consider a 2D multiply connected network domain $\mathcal{N} \in \mathbb{R}^2$ with boundaries $\partial\mathcal{N} = \gamma_0 - \gamma_1 - \cdots - \gamma_l$, where $\gamma_0$ is the outer boundary and $\gamma_i$, $1 \le i \le l$, are the hole boundaries.

Each iteration of the parametrization algorithm contains $l$ steps: in the first step, we conformally map $\gamma_0$ to the unit circle and $\gamma_1$ to a concentric circle. Let $\phi_1 : \mathcal{N} \to \mathbb{D}$ be the conformal map of the first step, where $\mathbb{D}$ is the unit disk. Then in the $i$-th step, $i > 1$, we conformally map $\phi_{i-1} \circ \cdots \circ \phi_1(S)$ to the unit disk $\mathbb{D}$ such that $\gamma_0$ is mapped to the unit circle and $\gamma_i$ to a concentric circle, i.e., $\phi_i : \phi_{i-1} \circ \cdots \circ \phi_1(\mathcal{N}) \to \mathbb{D}$. Note that $\phi_1(\gamma_1)$ is a circle. After the mapping, $\phi_2 \circ \phi_1(\gamma_1)$ is still close to a circle.

Intuitively, all the boundaries are getting rounder and rounder in the iterations, and eventually become circles. As each map $\phi_i$ is conformal, Henrici's theorem (see [7], pp.502-505) on complex analysis guarantees the convergence of our method.

**3D WSN.** The proof for volumetric case is similar to the proof of planar case. Each time we map an inner void to the center spherical void, the mapping can be extended to the volume with its reflection with respect to the boundary surface of the inner void. Eventually, the complement of the union of reflected volumes is a Cantor set, with zero measure. The harmonic map can be extended to the whole volume with all voids filled. By using the Poisson integral formula for harmonic maps [20], it can be shown that the images of the inner boundary surfaces are spherical.

## References


1. Bose, P., Morin, P., Stojmenovic, I., & Urrutia, J. (1999). Routing with guaranteed delivery in Ad Hoc wireless Networks. In *Proceedings of the 3rd ACM DialM*.
2. Durocher, S., Kirkpatrick, D., Narayanan, L. (2008). Guaranteed delivery in three-dimensional Ad Hoc wireless networks. In *Proceedings of the 9th ICDCN*.
3. Fang, Q., Gao, J., Guibas, L. (2004). Locating and bypassing holes in sensor networks. In *Proceeding of the 23rd IEEE INFOCOM*.
4. Flury, R., Wattenhofer, R. (2008). Randomized 3D Geographic Routing. In *Proceedings of the 27th IEEE INFOCOM*.
5. Gao, J., Guibas, L., Hershberger, J., Zhang, L., Zhu, A. (2003). Geometric spanner for routing in mobile networks. In *Proceedings of the 14th ACM MobiHoc*.
6. Hamilton, R. (1982). Three-Manifolds with positive Ricci curvature. *Elsevier Journal of Differential Geometry*, *17*(2), 255–306.
7. Henrici, P. (1993). *Applide and computational complex analysis, discrete fourier analysis, Cauchy integrals, construction of conformal maps, univalent functions* (Vol. 3). Hoboken: Wiley-Interscience.
8. Karp, B., Kung, H. (2000). GPSR: Greedy Perimeter Stateless Routing for Wireless Networks. In *Proceeidngs of the 6th ACM MobiCom*.
9. Kempe, D., Dobra, A., Gehrke, J. (2003). Gossip-based computation of aggregate information. In *Proceedings of the 44th IEEE FOCS*.
10. Kleinberg, R. (2007). Geographic routing using hyperbolic space. In *Proceedings of the 26th IEEE INFOCOM*.
11. Leighton, T., Moitra, A. (2008). Some results on greedy embeddings in metric spaces. In *Proceedigns of the 49th IEEE FOCS*.
12. Levis, P., Lee, N., Welsh, M., Culler, D. (2003). TOSSIM: accurate and Scalable Simulation of Entire TinyOS Applications. In *Proceedings of the 1st ACM SenSys*.
13. Li, F., Luo, J., Zhang, C., Xin, S., He, Y. (2011). UNFOLD: UNiform fast on-line boundary detection for dynamic 3D wireless sensor networks. In *Proceedings of the 12th ACM MobiHoc* (pp. 141–152).
14. Li, F., Zhang, C., Luo, J., Xin, S., & He, Y. (2014). LBDP: localized boundary detection and parametrization for 3-D sensor networks. *IEEE/ACM Transactions on Networking*, *22*(2), 567–579.
15. Li, J., Jannotti, J., Decouto, D., Karger, D., Morris, R. (2001). A scalable location service for geographic Ad-Hoc routing. In *Proceedings of the 7th ACM MobiCom*.
16. Luo, J., Li, F., He, Y. (2011). 3DQS: Distributed data access in 3D wireless sensor networks. In *Proceedings of IEEE ICC* (pp 1–5).
17. Papadimitriou, C., & Ratajczak, D. (2005). On a conjecture related to geometric routing. *Elsevier Theoretical Computer Science*, *344*(1), 3–14.
18. Rao, A., Ratnasamy, S., Papadimitriou, C., Shenker, S., Stoica, I. (2003). Geographic routing without location informtion. In *Proceedings of the 9th ACM MobiCom*.
19. Sarkar, R., Yin, X., Gao, J., Luo, F., Gu, X. (2009). Greedy routing with guaranteed delivery using Ricci flows. In *Proceedings of IPSN '09*.
20. Schoen, R., & Yau, S.-T. (1997). *Lectures on Harmonic Maps* (Vol. 2). Cambridge, MA: International Press.
21. Xia, S., Yin, X., Wu, H., Jin, M., Gu, X. (2011). Deterministic greedy routing with guaranteed delivery in 3D wireless sensor networks. In *Proceedings of the 12th ACM MobiHoc*.
22. Zeng, W., Sarkar, R., Luo, F., Gu, X., Gao, J. (2010). Resilient routing for sensor networks using hyperbolic embedding of universal covering space. In *Proceedings of the 29th IEEE INFOCOM*.
23. Zeng, W., Yin, X., Zhang, M., Luo, F., Gu, X. (2009). Generalized Koebe's method for conformal mapping multiply connected domains. In *SIAM/ACM SPM*.
24. Zhang, C., Luo, J., Xiang, L., Li, F., Lin, J., He, Y. (2012). Harmonic Quorum Systems: Data Management in 2D/3D Wireless Sensor Networks with Holes. In *Proceedings of the 9th IEEE SECON* (pp. 1–9).


**Minqi Zhang** received the Bachelor Degree from the School of Computer Engineer of the Tianjin University, in 2010. He is a Ph.D. student in the School of Computer Engineer at Nanyang Technological University, Singapore. His research interests are computational conformal geometry and its application to a wide range of fields, include the surface parameterization, surface registration for medical image, network routing, etc

**Feng Li** received his B.S. degree from Shandong Normal University, China, in 2007, and the M.S. degree from Shandong University, China, in 2010. He is currently a Ph.D. student at the School of Computer Engineering, Nanyang Technological University, Singapore. His research interests are computational geometry and its application in wireless sensor networks.

**Ying He** received the B.S. and M.S. degrees in Electrical Engineering from Tsinghua University, China, and the Ph.D. degree in Computer Science from the State University of New York (SUNY), Stony Brook, USA. He is currently an associate professor at the School of Computer Engineering, Nanyang Technological University, Singapore. His research interests are in the broad areas of visual computing, with a focus on the problems that require geometric computation and analysis. More information can be found at http://www.ntu.edu.sg/home/yhe.

**Juncong Lin** is now an associate professor at Xiamen University. He received the Ph.D. degree in Computer Science from Zhejiang University. His research interests include geometry processing and sketch based modeling.

**Xianfeng Gu** received the Ph.D Degree in Computer Science from Harvard University in 2003. He is an associate professor of computer science and the director of the 3D Scanning Laboratory in the Department of Computer Science at Stony Brook University, New York. His research interests include computer vision, graphics, geometric modeling, and medical imaging. His major works include global conformal surface parameterization in graphics, tracking and analysis of facial expression in vision, manifold splines in modeling, brain mapping and virtual colonoscopy in medical imaging, and computational conformal geometry. He won the US National Science Foundation CAREER Award in 2004. He is a member of the IEEE.

**Jun Luo** received his B.S. and M.S. degrees in Electrical Engineering from Tsinghua University, China, and the Ph.D. degree in Computer Science from EPFL (Swiss Federal Institute of Technology in Lausanne), Lausanne, Switzerland. From 2006 to 2008, he has worked as a post-doctoral research fellow in the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, Canada. In 2008, he joined the faculty of the School of Computer Engineering, Nanyang Technological University in Singapore, where he is currently an assistant professor. His research interests include wireless networking, mobile and pervasive computing, distributed systems, multimedia protocols, network modeling and performance analysis, applied operations research, as well as network security. More information can be found at http://www3.ntu.edu.sg/home/junluo.