International Journal of Geographical Information Science
Vol. 00, No. 00, Month 2012, 1–22

Research Article

Decentralized and coordinate-free computation of critical points and surface networks in a discretized scalar field

Myeong-Hun Jeong[a] *, Matt Duckham[a], Allison Kealy[a],
Harvey J. Miller[b], Andrej Peisker[a]

[a] *University of Melbourne, VIC, Australia*; [b] *University of Utah, Utah, US*

(*Received 00 Month 200x; final version received 00 Month 200x*)

This paper provides a decentralized and coordinate-free algorithm, called DGraF (decentralized gradient field) to identify critical points (peaks, pits, and passes) and the topological structure of the surface network connecting those critical points. Algorithms that can operate in the network without centralized control and without coordinates are important in emerging resource-constrained spatial computing environments, in particular geosensor networks. Our approach accounts for the discrepancies between finite granularity sensor data and the underlying continuous field, ignored by previous work. Empirical evaluation shows that our DGraF algorithm can improve the accuracy of critical points identification when compared with the current state-of-the-art decentralized algorithm, and matches the accuracy of a centralized algorithm for peaks and pits. The DGraF algorithm is efficient, requiring $O(n)$ overall communication complexity, where n is the number of nodes in the geosensor network. Further, empirical investigations of our algorithm across a range of simulations demonstrate improved load balance of DGraF when compared with an existing decentralized algorithm. Our investigation highlights a number of important issues for future research on the detection of holes and the monitoring of dynamic events in a field.

Keywords: geosensor network; decentralized spatial computing; surface networks; critical points; discrete Morse theory

1. Introduction

Geosensor networks (wireless sensor networks that monitor physical phenomena in geographic space, Nittel 2009) are having a broad impact across a wide range of applications

*Corresponding author. Email: mh.jeong@student.unimelb.edu.au

ISSN: 1365-8816 print/ISSN 1362-3087 online
© 2012 Taylor & Francis
DOI: 10.1080/1365881YYxxxxxxxx
http://www.informaworld.com

of geospatial data, from habitat monitoring to vehicle tracking (Estrin *et al.* 2002, Akyildiz *et al.* 2002, Chong and Kumar 2003). In some of these domains, there is a need to monitor *scalar fields*, such as sea temperature and salinity (Leonard *et al.* 2007), noise or airborne pollution (Santini *et al.* 2008, Murty *et al.* 2008), or density of mobile phone activity (Horanont and Shibasaki 2008). This paper presents and tests an efficient and accurate algorithm, DGraF (decentralized gradient field), for characterizing scalar fields through the identification of critical points (pits, peaks, and passes) and the surface networks that connect those points.

The paper addresses two key challenges that arise in the design of our algorithm. First, geosensor networks are highly resource-constrained computing environments. Limited energy resources mean algorithms must typically be decentralized, operating in the geosensor network with no centralized control and relying solely on local communication between nearby (one-hop) neighbors (Krishnamachari *et al.* 2002, Madden *et al.* 2002, Akkaya *et al.* 2008). In a decentralized system, no node possesses global knowledge of the system state (Lynch 1996). Consequently, centralized algorithms are usually not directly applicable in a decentralized computing environment, except by reverting to unscalable communication of all data from across the entire network back to a single, centralized sink node. Energy, cost, and practical limitations also favor algorithms that are able to operate without requiring accurate coordinate positioning information. Positioning technology, such as GPS, places a substantial drain on constrained resources; increases the cost of sensors, so decreasing the size of affordable networks; and cannot position ubiquitously across a wide range of environments (e.g., GPS in indoor environments or under dense vegetation cover). Our DGraF algorithm is both decentralized and does not require any coordinate positions for nodes.

Second, the classical models of surface networks are founded on important assumptions about the continuity of the underlying field. The finite spatial granularity of sensor data effectively violates this assumption, and makes the identification of critical points nontrivial. We provide a proof of the inadequacy of discretizations of continuous fields to capture the critical points those fields, an issue that previous works have not addressed. These problems of the fidelity of a discretization led us to extend the standard definitions of critical points in discretized fields, and investigate empirically the accuracy of the critical point identification process. Our new technique uses a "face-based" approach to critical point identification, embedded in an algorithm that we demonstrate improves both efficiency and accuracy when compared with the state-of-the-art decentralized alternative.

Section 2 discusses in more detail the limitations of previous research on surface networks, and the problems posed by finite spatial granularity. Further exploring the problems of granularity, Section 3 shows that a discretization can provide neither necessary nor sufficient evidence of critical points in its underlying continuous surface. Responding to these limitations, extended definitions of pit, peak, and pass are used in Section 4 as the basis of a new decentralized algorithm for identifying critical points and surface networks in a scalar field. Section 5 presents an experimental evaluation of the efficiency (communication complexity) and accuracy of the algorithms using simulations. These experiments are discussed in Section 6. Section 7 concludes with a summary and suggestions for future work.

2. Background

The description of spatial fields has long been of both fundamental theoretical interest and direct application to a wide range of areas, including terrain surfaces, population density, temperature surfaces, and so forth. Maxwell (1870) intuitively described the Earth's surface as hills and dales. He also proposed relations between the number of summits, passes, immits, and bars. In a later approach based on differential topology, Morse theory developed effective tools to capture the topological structure of surface networks (Smale 1960, Milnor 1969, Matsumoto 2002).

Specifically, Morse theory deals with the relationship between functions defined on the space and the shape of the space. A surface can be represented by a function $z = f(x, y)$, where (x, y) is a point in the plane, and z is the "height" or intensity at that point. If a point p satisfies $\frac{\partial f}{\partial x}(p) = 0$ and $\frac{\partial f}{\partial y}(p) = 0$, p is called a *critical point* of the function f. Based on the behavior of the function f, these critical points are classified as peaks, pits, and passes, and are an intuitive and sufficient qualitative representation of the surface. The extracted critical points can be connected by critical lines (i.e., a surface network coined by Pfaltz 1976). For example, ridges are lines connecting peaks and passes; channels (valleys) are lines connecting passes and pits.

The transition from continuous surfaces to discrete data structures and algorithms has been an active research area, with at least five different broad approaches to discretization evident in the literature: triangulations; piecewise linear functions; contour trees and Reeb graphs; discrete Morse theory; and morphometric analysis.

First, Takahashi *et al.* (1995), Bajaj and Schikore (1998) compute surface networks by comparing the height and geometric characterization of vertices in a *triangulation*. They extract the critical points from the triangular mesh, and then trace the surface network that connects peaks and pits through passes.

An alternative to triangulations, Banchoff (1970) defines critical points based on a *piecewise linear functions* on polyhedral surface embedded in $\mathbb{R}^3$. This approach has been exploited by Edelsbrunner *et al.* (2003), Danovaro *et al.* (2006) for generating a Morse-Smale complex whose 1-skeleton is a surface network.

Other data structures based on Morse theory include *contour trees* (Kreveld *et al.* 1998, Carr *et al.* 2003, Chiang and Lu 2005) and *Reeb graphs* (Shinagawa *et al.* 1991, Biasotti *et al.* 2004, Edelsbrunner *et al.* 2004). While surface networks can be obtained by connecting the identified critical points on the boundary of a continuous field, contour trees and Reeb graphs represent fields by analyzing the evolution of the level sets, in terms of the appearance, disappearance, joining, and splitting of connected components. Reeb graphs can describe these topological features of all level sets, but contour trees can only represent the evolution of simply-connected domains (i.e., loop-free Reeb graphs, Biasotti *et al.* 2008).

Fourth, Forman (1998, 2002) proposed *discrete* Morse theory, which uses a discrete Morse function to assign a single number to each cell of a simplicial complex. This model can represent the topology of a surface without relying upon any continuous functions. This tool has a crucial role in a variety of combinatorial and topological problems (Lewiner *et al.* 2004, Ayala *et al.* 2010, Reininghaus *et al.* 2010).

Finally, Schneider and Wood (2004) and Wood and Rana (2000) use morphometric analysis to fit an approximating function to the given set of (grid) points, differentiating the function to identify critical points. Unlike the other approaches discussed above, these two techniques do not rely an explicit cell complex. However, they still require a grid configuration of points, itself an implicit form of cell complex.

All these approaches rely on strong assumptions about the existence of explicit or implicit cell complexes. However, explicit cell complexes are frequently inefficient or impractical to construct in the case of geosensor networks. The communication constraints of decentralized computation (limited communication range and capacity) mean that cell complexes, such as triangulations, that may be efficiently computed in a centralized system, are not scalable in a decentralized environment (Li *et al.* 2002, Alzoubi *et al.* 2003). Similarly, implicit cell complexes, such as highly regular grid deployments of sensors, can only rarely be supported in environmental geosensor networks in practice.

Other common limitations of geosensor networks—such as no, or low-cost positioning systems—mean that coordinate position or quantitative range- or direction-finding information is frequently unreliable or absent. The approaches above rely on information about distances or directions between nodes, and frequently on coordinate position, either directly or indirectly in the construction of appropriate cell complex (e.g., the construction of a triangulation).

Finally, all the approaches discussed above assume centralized computation, where all information from the network is first communicated to, and collated by a central sink node before the surface network and critical points can be computed. Consequently, these algorithms are ill-suited to efficient operation in a distributed wireless geosensor network. In such contexts, decentralized algorithms, able to identify critical points in the network and based only on limited and short-range communication of information, are required. Two key papers that have already tackled the problem of the identification of critical points in a spatial field monitored by a decentralized and coordinate-free geosensor network are Sarkar *et al.* (2008) and Zhu *et al.* (2009). Both papers are similar in their basic approach to detection of a peak or pit: a peak (pit) has neighbors whose value is greater (smaller) than itself (cf. Duckham 2013). While Sarkar *et al.* (2008) use a sweep algorithm (see Skraba *et al.* 2006) to identify passes, Zhu *et al.* (2009) attach the problem by adopting a Morse-Smale decomposition based on gradient vectors. However, a major assumption in these state-of-the-art algorithms is that the spatial field monitored by a geosensor network is relatively smooth. The accuracy of these algorithms degrades rapidly as the field becomes rougher. Both algorithms also assume that nodes are distributed evenly and densely across space. The approaches tend to fail to correctly identify critical points as the spatial distribution of nodes within a geosensor network becomes less even, and holes in the network coverage appear.

In summary, classical models of surface networks based on Morse theory are not well-adapted for use in geosensor networks, because they frequently rely on coordinate positions, or at least range- or direction-finding; are designed for centralized computation; and require a well-defined cell complex. Further, the state-of-the-art decentralized algorithms for critical point identification rely on strong assumptions about the distribution of nodes and granularity of the network relative to the field, and decrease rapidly in reliability as these restrictive assumptions are relaxed. Finally, previous work has implicitly assumed that the critical points in a dense discretization will necessarily reflect those in the original continuous surface. As we show in the following section, it is not in general possible to infer the properties of a continuous surface from its discretization, irrespective of the discretization density.

Consequently, this paper presents and tests a new, decentralized algorithm for identifying critical points in a monitored scalar field. Acknowledging that any discretization can never guarantee to faithfully represent a continuous surface, we demonstrate that our algorithm can still equal or outperform the existing state-of-the-art algorithms both in terms of reliability and efficiency.

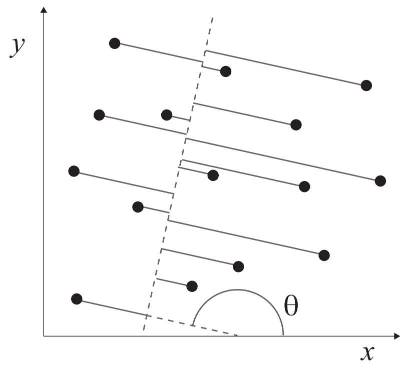


Figure 1. Construction of Lemma 3.2

3. Discrete surface networks

A key assumption made in previous work is that a suitably dense discretization of a continuous surface will preserve the properties of that surface. This expectation is made explicit in Takahashi *et al.* (1995, p. 181): “The techniques for extracting critical points are based on the assumption that the critical points can be extracted from the discrete data in the same way as from the continuous one. If the sample data is dense enough, we can extract correct critical points using the techniques.”

In this section we refute this expectation. It may already be clear to the reader that the critical points of a continuous surface are not necessarily preserved by discretization: adverse positioning or coarse granularity of sample points can always lead to finer-scale detail being omitted or distorted. However, here we also show the converse: a discretization that appears to contain a critical point is not sufficient to indicate that the underlying continuous surface, which that discretization samples, also contains a critical point. Importantly, we show this result holds *irrespective of the density of the points in the discretization.*

Definition 3.1: Let $f : \mathbb{R}^2 \to \mathbb{R}$ be a Morse function which represents a “true” continuous surface. Let discretization $D = \{(x_1, y_1), ..., (x_m, y_m)\}$ be an arbitrary, finite set of points in the xy-plane, $D \subset \mathbb{R}^2$.

First, we make use of a lemma that states it is always possible to draw parallel lines through a finite set of points such that each line intersects exactly one point.

Lemma 3.2: *For any data set D, there exists an angle θ, measured counterclockwise from the x-axis, such that a line drawn through any point $(x_i, y_i) \in D$ at angle θ will pass through no other point in D.*

Proof: Note that the number of non-allowed directions is finite. A discretization of size $|D| = m$ will generate at most $\frac{1}{2}m(m-1)$ non-allowed directions, since each pair of points in D defines at most one non-allowed direction. However, the number of choices of $\theta \in [0, 2\pi)$ is infinite. Hence, there must always exist some angle θ with the required property. □

Figure 1 shows the intuition behind this lemma: we can always find a direction that allows the points of D to be totally ordered.

Theorem 3.3: *Given a Morse function $f : \mathbb{R}^2 \to \mathbb{R}$, a discretization D, there exists a surface containing no critical points which exactly interpolates the discrete surface $D_f \subset \mathbb{R}^3$, the set of 3D points resulting from applying f to D, $D_f = \{(x_1, y_1, f(x_1, y_1)), ..., (x_m, y_m, f(x_m, y_m))\}$.*

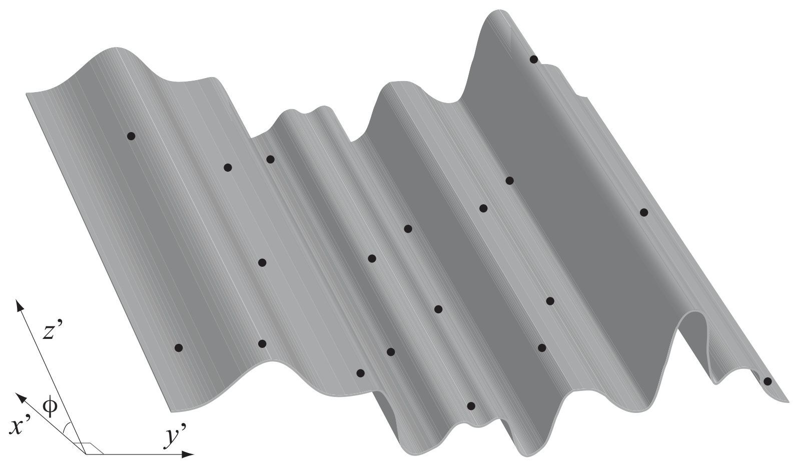


Figure 2. Illustration of "corrugated roof" surface with no critical points, fitted to arbitrary point set D

Proof: We define an auxiliary coordinate system (x', y', z') with arbitrary origin; x' axis oriented in the direction θ; y' axis normal to this axis along the (x, y) plane; and z' a skew axis with a component normal to the (x, y) plane. The z' axis is constructed via a rotation of vertical z axis through an arbitrary angle $\phi \in (0, \pi/2)$ about the positive y' axis. Now consider the projection of D_f onto the (y', z') plane. By construction, the y' co-ordinates of the projections must be distinct. The set of projected points $\{(y'_1, z'_1), ..., (y'_m, z'_m)\}$ may be interpreted as the lateral locations and "heights" above the y' axis as seen by an observer looking at the array of points $D_f = \{(x_1, y_1, z_1), ..., (x_m, y_m, z_m)\}$ in a direction normal to the (y', z') plane (i.e., at a slant with angle ϕ). Because the y'_i are disjoint, it follows that one can always specify an interpolating polynomial function $p : \mathbb{R} \to \mathbb{R}$ such that $p(y'_i) = z'_i$ for $i = 1, ..., m$. In general this would need to be a polynomial of degree at least m. Now by extruding this polynomial curve to infinity in a direction normal to the (y', z') plane we generate a surface S which (by construction) interpolates all $(x_i, y_i, z_i) \in D$. Let $g : \mathbb{R}^2 \to \mathbb{R}$ be a scalar function which generates S i.e. $S(x, y) = (x, y, g(x, y))$ everywhere (assuming a parametrisation of S by its projection onto the (x, y) plane). We claim that g is necessarily devoid of critical points since there will always be a non-zero component of the gradient field $\nabla g = (\frac{\partial g}{\partial x}, \frac{\partial g}{\partial y})$ which points up the incline of the (y', z') plane, i.e., $\nabla g \cdot \hat{x}' = \tan \phi$ for $\phi \in (0, \pi/2)$ where $\hat{x}'$ is a unit vector in the x' direction. It follows that ∇g is everywhere non-zero and therefore has no critical points since the necessary condition $\nabla g = 0$ is nowhere satisfied. This completes the constructive proof for the theorem. □

Intuitively, the proof fits to the discretization a surface with a shape akin to an inclined corrugated roof, whose "ripples" ensure it passes through each discrete point, and whose "slant" means it everywhere possesses non-zero gradient (and therefore possesses no critical points). Figure 2 illustrates the construction, showing the fitting of a "corrugated roof," inclined at angle ϕ, to arbitrary set of points.

In summary, while the shape of a discretized surface may on average reflect the underlying shape, the location or existence of critical points in a discretization are neither necessary nor sufficient indications of the location or existence of critical points in the underlying surface from which the discretization was derived.

4. Algorithms

From section 3 we conclude that a discretization of a continuous surface may suggest, but not strictly *imply*, the location or existence of critical points in the underlying continuous surface. This results has two direct consequences. First, the accuracy of algorithms for computing critical points from a continuous surface must be evaluated empirically (explored in Section 5). Second, that it is appropriate to revisit the basic definitions of critical points, and adapt them to the limited spatial granularity of a geosensor network (this section). These adapted definitions form the basis of our coordinate-free DGraF algorithm, capable of computing these revised definitions in a decentralized geosensor network. For ease of exposition the algorithm is separated into two parts: identifying peaks/pits (Algorithm **1**); and identifying passes/network topology (Algorithm **2**). In the following section we explore the performance of the algorithms, in particular the accuracy of the identified critical points when compared with the underlying continuous surface.

4.1. *Basic definitions*

Following previous work (Duckham 2013), a geosensor network can be modeled as an connected, undirected graph $G = (V, E)$, where V is the set of sensor nodes and $E \subseteq V \times V$ is the set of direct, one-hop communication links between neighboring nodes. We refer to the (communication) neighborhood of each node using the function $nbr : V \to 2^V$, where $nbr(v) = \{v' | \{v, v'\} \in E\}$. Each node can sense the scalar field (z-value) at its immediate location, modeled as a function $sense : V \to \mathbb{R}$. Further, each node is assumed to possess an identifier, represented by the function $id : V \to \mathbb{N}$.

Previous research (Lewiner *et al.* 2004, Reininghaus *et al.* 2010, Lewiner 2012) in computer graphics address the topology of discrete scalar fields based on triangulated irregular networks (TINs). They use a gradient vector and a discrete Morse function to identify critical features. Unfortunately, decentralized construction of a triangulation in a sensor network is known to be inefficient (Li *et al.* 2002, Alzoubi *et al.* 2003). This inefficiency results because the physical communications network may not contain a triangulated subgraph. Instead, neighbors in a triangulation of sensor nodes may be separated by arbitrarily many hops in the physical communication network, leading to considerable communication overheads finding and maintaining connections between triangulation neighbors.

Previous work has already defined a peak (or pit) as below, after (Sarkar *et al.* 2008, Zhu *et al.* 2009, Duckham 2013), without requiring a triangulation.

Definition 4.1: A *peak* is a node v such that all neighbors of v sense a lower value, $\max_{v_i \in nbr(v)}(sense(v_i)) < sense(v)$. Similarly, a *pit* is a node v such that all neighbors of v sense a higher value, $\min_{v_i \in nbr(v)}(sense(v_i)) > sense(v)$.

For nodes that are neither peaks (nor pits), Zhu *et al.* (2009) further define ascent (or descent) vectors as follows:

Definition 4.2: The *ascent vector* of a non-peak node v (written $av(v)$) is the (unique) directed edge (v, v') where v' is the neighbor with the maximum sensed value, $\max_{v_i \in nbr(v)}(sense(v_i))$. Similarly, the *descent vector* $dv(v)$ of a non-pit node v is the (unique) directed edge (v, v') where v' is the neighbor with the minimum sensed value, $\min_{v_i \in nbr(v)}(sense(v_i))$.

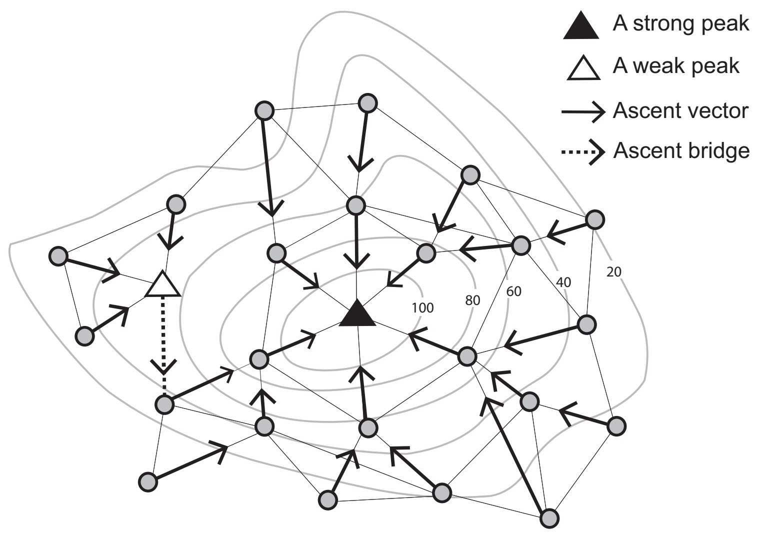


Figure 3. The identification of a peak. A filled triangle is a strong peak and a non-filled triangle is a weak peak. Arrows indicate ascent vectors, with the ascent bridge highlighted as a dashed arrow.

In this paper we refine these existing definitions by distinguishing a *strong* peak (or pit):

Definition 4.3: A *strong* peak is a node v such that v is a peak and the ascent vectors of all the neighbors of v point to v. Similarly, a *strong* pit is a node v such that v is a pit and the descent vectors of all the neighbors of v point to v. A peak (or pit) that is not a strong peak (or pit) is called a *weak* peak (or pit).

For example, in Figure 3 the strong peak (filled triangle symbol) has neighbors that have lower values than itself (definition 4.1), and all the neighbors' ascent vectors (definition 4.2) point to the peak (definition 4.3). By contrast, the (weak) peak (non-filled triangle symbol) has neighbors whose values are lower than itself, but one of those neighbor's ascent vectors does not point to that peak.

This phenomenon can occurs frequently in a sensor network as a result of limited network granularity and connectivity. Thus an important difference in our work to previous work is that we require (strong) peak and pit nodes to compare with not only their neighbors' sensed values but also their neighbors ascent vectors (i.e., their neighbors' sensed values). This is the essence of our "face-based" approach, for robust identification of critical points in a geosensor network.

It should be noted that we do not claim that the existence of a weak peak (pit) is sufficient to guarantee the absence of a peak (pit) in the underlying continuous surface, any more that we would claim that the existence of a strong peak (pit) is sufficient to guarantee a corresponding peak (pit) in the underlying continuous surface (cf. Section 3). Greater accuracy might be achieve by further extending our definitions, iterating the search for weak peaks and pits beyond one-hop neighbors, so constructing n-hop strong peak and pits. However, we will demonstrate empirically that our minimal definitions of weak peaks and pits is are an efficient and accurate heuristic that, on average, achieves a good balance: discounting many spurious critical points at the same time as relatively infrequently discounting true critical points.

Assuming every non-peak (non-pit) node has a unique ascent (descent) vector (i.e., no two neighbors have identical maximum/minimum sensed values, see definition 4.2), the set of ascent (set of descent) vectors for a sensed spatial field forms a forest (set of rooted trees). However, some of these trees' roots will be weak peaks (or pits). To address this we need to build *bridges* from the weak peaks (pits) to strong peaks (pits) as follows:

Definition 4.4: An ascent *bridge* is a (unique) directed edge $(v, v') \in G$ such that: i. v is a weak peak; ii. v' has ascent vector (v', v'') that points away from v, i.e., $v'' \neq v$; iii. v' has a higher sensed value $sense(v')$ than any other neighbor of v that satisfies conditions i. and ii. Similarly, a descent *bridge* is the (unique) directed edge (v, v') such that v is a weak pit, $v' \in B$ where $B = \{v' \in nbr(v) | dv(v') \neq v\}$, and v' has the minimum sensed value, $\min_{v_i \in B}(sense(v_i))$.

Now for any node we can identify the unique strong peak (strong pit) associated with that node, much as we might identify catchments in conventional, continuous spatial fields.

Definition 4.5: The set of all ascent vectors and ascent bridges forms a forest F_a of rooted trees. The terminal strong peak of a node v, $tspeak(v)$, is the root of that tree $T_a \in F_a$ that contains the node v. Similarly, the set of all descent vectors and descent bridges forms a forest F_d of rooted trees. The terminal strong pit of a node v, $tspit(v)$, is the root of that tree $T_d \in F_d$ that contains the node v.

Finally, we are now ready to define a discrete version of a pass:

Definition 4.6: A *pass-edge* is an undirected edge $\{v_1, v_2\}$ where v_1 and v_2 have different terminal strong peaks and pits, $tspeak(v_1) \neq tspeak(v_2)$ and $tspit(v_1) \neq tspit(v_2)$.

A pass-edge is analogous to a pass in classical Morse theory. The detection of passes in a geosensor network without coordinate information is a key challenge in the previous work of (Sarkar *et al.* 2008, Zhu *et al.* 2009). Using the presence of divergent terminal strong peaks/pits associated with pairs of neighboring nodes to identify passes allows our algorithm to more efficiently and accurately identify passes decentrally.

Figure 4 provides an example of a pass-edge (pass), shown as a pair of neighboring square nodes with different terminal strong pits and peaks.

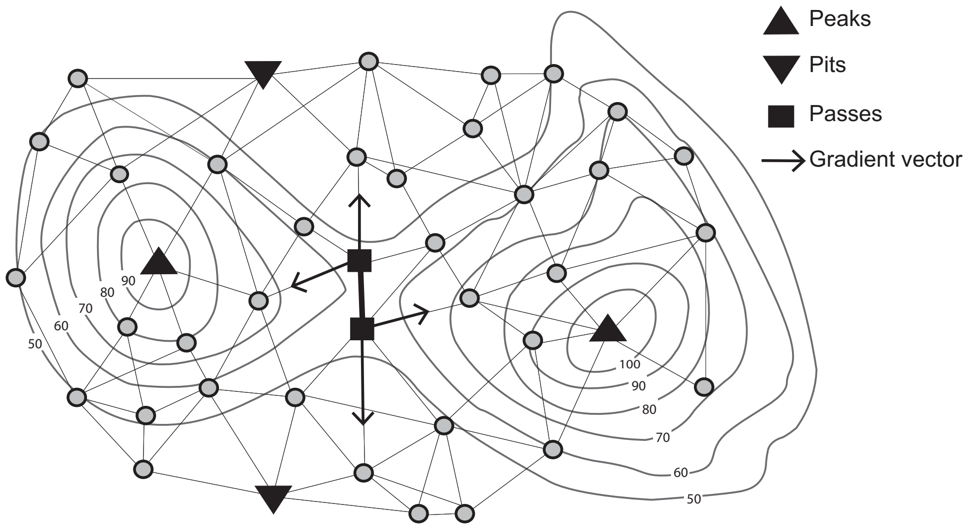


Figure 4. The identification of pass-edge (pass).

4.2. *The identification of peaks/pits*

Algorithm **1** provides our decentralized protocol for identification of strong peaks and pits, the first component of DGraF. The algorithm specification style follows that presented in Santoro (2007), Duckham (2013). After the restrictions (assumptions) listed in the header to the algorithm, the algorithm is structured hierarchically, first by states (in small caps), then by events, then by unique actions (sequences of atomic operations) that are triggered in response to state/event pairs. In brief, the algorithm proceeds as follows:

- In a INIT state, each node broadcasts a `ping` message to neighbors (line 6). Using this information, each node can deduce its ascent/descent vector, and whether it is a peak or pit.
- Peaks (pits) initiate a `tcel` (`bcel`) message to neighbors in order to check whether they are strong peaks (pits), i.e., line 19. A node in an IDLE state receiving a `tcel` (`bcel`) message from a potential peak (pit) returns a message that confirms or denies if its ascent (descent) vector points to that peak (pit), i.e., line 20–25.
- A peak (pit) that receives positive confirmation messages from all its neighbors is a strong peak (pit). Strong peaks (but not strong pits) initiate a top-down `invt` invitation message to its neighbors (lines 30–34).
- A weak peak will wait for a strong peak information (line 27) to cascade down to it.

The identification of strong pits follows a similar pattern of identifying strong peaks, initiated when top-down invitation messages reach pits. For brevity, the process for the PITX state has been omitted from Algorithm **1**.

4.3. *The identification of passes*

Algorithm **2** presents the detection of passes. First, each node can deduce its (strong) peak through top-down invitation messages from every strong peak. When top-down invitation messages reach a strong pit, these nodes initiate a bottom-up invitation message. Each node therefore can recognize its neighbors' associated peak and pit based on the neighbors' ascent and descent vectors (see Algorithm **2**, lines 6 and 16). As a result, if the end points of edges are identified with two different peaks and pits, these edges are termed pass-edges (see Algorithm **2**, line 12 and line 24). For example, each node can compare a peak identifier through top-down invitation messages. If peak identifiers are different between two neighboring nodes, this edge can be regarded as a channel. Conversely, if pit identifiers are different between neighboring nodes, this edge will be a ridge. If the neighbors that make up an edge have differ in both peak and pit identifiers, that edges is a pass-edges. The required information can be deduced from passes' associated peak and pit information (see Algorithm **2**, lines 25, and 30). This identification corresponds to the cut locus in Zhu *et al.* (2009). Algorithm **2** therefore can capture the topological structure of the surface network (i.e., Morse-Smale complex).

Depending on the network connectivity, it is possible that multiple pass-edges may be identified for the same two peak/pit pairs. In such cases, neighboring pass-edges are grouped together based on comparison of information about their associated peaks and pits (see Algorithm **2**, lines 27 and 32).

Algorithm 1 DGraF: The identification of peaks

```
1: Restrictions: Geosensor network, $G = (V, E)$; sensor function $sense : V \to \mathbb{R}$; communication
   neighborhood $nbr : V \to 2^V$; identifier function $id : V \to \mathbb{N}$; reliable communication.
2: State Trans. Sys.: $\langle\{$INIT, LSTN, IDLE, PEAK, PITX$\}, \{($INIT, LSTN$), ($LSTN, IDLE$), ($LSTN, PEAK$),
   ($LSTN, PITX$)\}\rangle$
3: Initialization: all nodes in state INIT.
4: Local variables: list of upstream neighbor identifiers, Nu, initialized empty; list of downstream
   neighbor identifiers, Nd, initialized empty; list of neighbor identifiers of top-down invitation
   messages, Nt, initialized empty; an ascent vector, av, initialized empty; a descent vector, dv,
   initialized empty; list of edge, Edge, initialized empty; list of top-cells adjacent to a peak,
   Tcells, initialized empty; list of a strong peak id, Pkid, initialized empty; list of weak peak
   ID, Wpeaks, initialized empty
INIT
 5: Spontaneously
 6:     broadcast (ping, $\mathring{sense}$, $\mathring{id}$)
 7:     become LSTN
LSTN
 8: Receiving (ping, $h$, $i$)
 9:     $Edge := Edge \cup \{\mathring{id}, i\}$                          --Save edge information
10:     if $\mathring{sense} < h$ then
11:         $Nu := Nu \cup \{i, h\}$                          --Add i into upstream neighbor nodes
12:     if $\mathring{sense} > h$ then
13:         $Nd := Nd \cup \{i, h\}$                          --Add i into downstream neighbor nodes
14:     if $|Nu| + |Nd| =$ the number of neighbors then
15:         $av := \max\{Nu\}$ and $dv := \min\{Nd\}$          --Set ascent and descent vectors
16:         become IDLE
17:         if $|Nu| = 0$ then
18:             become PEAK
19:             broadcast (tcel, $\mathring{id}$)      --Broadcast top-cell messages to identify a strong peak
IDLE
20:     Receiving (tcel, $i$)                              --Receiving top-cell messages
21:         if $\mathring{av} = i$ then
22:             send (ctce, $\mathring{id}$, true) to a node with identifier $i$     --Send a true confirmation message
23:         else
24:             send (ctce, $\mathring{id}$, false) to a node with identifier $i$    --Send a false confirmation message
25:             $Wpeaks := Wpeaks \cup \{i\}$                   --Set an ascent bridge
PEAK
26:     Receiving (ctce, $i$, $bflag$)
27:         if $bflag =$ false then
28:             wait strong peak information from an ascent bridge
29:         else
30:             if i $\notin Tcells$ then
31:                 $Tcells := Tcells \cup \{i\}$          --Receive confirmation messages from neighbors
32:             if $|Tcells| = |Nd|$ then
33:                 $Pkid := Pkid \cup \{\mathring{id}\}$
34:                 broadcast (invt, $\mathring{id}$, $Pkid$)          --Broadcast invitation messages
PITX
35:     Events and actions follow a similar pattern to the PEAK state
```

4.4. *Scalability*

Examining Algorithms **1** and **2**, we observe that each node in the network must transmit exactly one `ping` message for network initialization and two invite (one top-down `invt`, one bottom up `binv`) messages for joining identifying their peak/pit "catchments." Further, peaks and pits must send one message to check for strong/weak peak/pit status (`tcel`, `bcel`) and neighbors of peaks and pits must send one confirmation message in

Algorithm 2 Identifying pass-edges (i.e., passes)

```
1: Fragment extends: Algorithm 1
2: Local variables: list of neighbor identifiers of top-down invitation messages, $Nt$, initialized
   empty; list of neighbor identifiers of bottom-up invitation messages, $Nb$, initialized empty; list
   of pass-edges (passes), $Cedges$, initialized empty; list of pass cluster, $Passcluster$, initialized
   empty; list of channels, $Channels$, initialized empty; list of ridges, $Ridges$, initialized empty
IDLE
3: Receiving (invt, $i$, $p$)                                  --Receiving top-down invitation messages
4:   $Nt := Nt \cup \{i\}$                                                      --No duplication
5:   if $av = i$ then                                                   --To define nodes' peak
6:     $Pkid := Pkid \cup \{p\}$
7:   if $|Nt| = |Nu|$ then
8:     broadcast (invt, $\mathring{id}$, $Pkid$)
9:   if $|Nt| = |Nu| + |Nd|$ then
10:    while $Edge \neq \emptyset$ do
11:      if The peaks of end points between edges are different then
12:        $Cedges := Cedges \cup Edge$                                   --Potential pass-edges
13: Receiving (binv, $i$, $p$)                                --Receiving bottom-up invitation messages
14:   $Nb := Nb \cup \{i\}$                                                     --No duplication
15:   if $dv = i$ then                                                    --To define nodes' pit
16:     $Ptid := Ptid \cup \{p\}$
17:   if $|Nb| = |Nd|$ then
18:     broadcast (binv, $\mathring{id}$, $Ptid$)
19:   if $|Nb| = |Nd| + |Nu|$ then
20:     while $Edge \neq \emptyset$ do
21:       if $|Cedges| > 1$ then                       --Potential pass-edges with different peaks
22:         while $Cedges \neq \emptyset$ do
23:           if The end points of pass-edge have the same pit information then
24:             $Cedges := Cedges - Edge$                 --Passes have different peaks and pits
25:             $Channels := Channels \cup Edge$              --Different peaks and the same pit
26:         $Passcluster := Passcluster \cup Cedges$       --To cluster a group of pass-edges as one
27:         broadcast (pcel, $\mathring{id}$, $Passcluster$)        --Broadcast a pass cluster message
28:       else
29:         if The end points of edges have the different pit information then
30:           $Ridges := Ridges \cup Edge$                   --The same peak and different pits
31: Receiving (pcel, i, passcluster)
32:   $Passcluster := Passcluster \cup$ passcluster                            --No duplication
33:   if The information of pass cluster is changed then
34:     broadcast (pcel, $\mathring{id}$, $Passcluster$)
```

response (`ctce`). Finally, a small number of one-hop messages will be required in some cases to exchange associated peak and pit identifiers at pass-edges (`pcel`).

In total, then, we expect the algorithm to generate $3|V| + m$ messages: exactly 3 `ping`, `invt`, `binv` messages per node, plus a small number m of additional `tcel`, `bcel`, `ctce`, `pcel` messages, where m is closely related to the number of detected peaks, pits, and passes. In most cases it can be assumed that $m \ll |V|$. Consequently, it is to be expected that overall communication complexity of our algorithm is linear, $O(n)$, where $n = |V|$ is the number of nodes in the network.

The following section, 5, investigates experimentally this and other analytical expectations made in this section.

5. Experiments

The DGraF algorithm was evaluated with respect to three primary features: overall scalability, load balance, and accuracy. For benchmarking, we compared the performance of DGraF with the state-of-the-art decentralized algorithm (Zhu *et al.* 2009) and a classic centralized alternative (Takahashi *et al.* 1995).

5.1. *Experimental setup*

Random differentiable scalar functions over the unit square, $f : [0, 1]^2 \to \mathbb{R}$, were used to generate a range of simulated surfaces using Matlab (MATLAB 2010). Takahashi *et al.*'s algorithm was used to identify critical points based on dense Delaunay triangulation of the surfaces. These critical points formed our "ground truth" for evaluating algorithm accuracy.

The surfaces varied in roughness, classified into four levels. In the case of level 1 and 2 surfaces, the functions took the form of quadric polynomials. Levels 3 and 4 surfaces were generated in a similar fashion as quadric polynomials of harmonic functions, i.e., $f(\sin(\omega x), \cos(\omega y))$, where $\omega = 4$ and 8 for level 3 and 4 surfaces respectively. As a guide, level 1 surfaces had on average 3–4 critical points; level 2, an average of 6 critical points; level 3, 8 critical points; and level 4, on average 36 critical points.

These surfaces were exported from MATLAB and loaded into an agent-based simulation system, Netlogo (Willensky 1999). Geosensor networks at five sizes, ranging from 1,000 to 16,000 nodes, were simulated in two different configurations, regular (grid) and irregular (random) node locations. The network was connected by the unit disk graph (UDG). The level of connectivity was kept constant across the different network sizes in order to guarantee comparability.

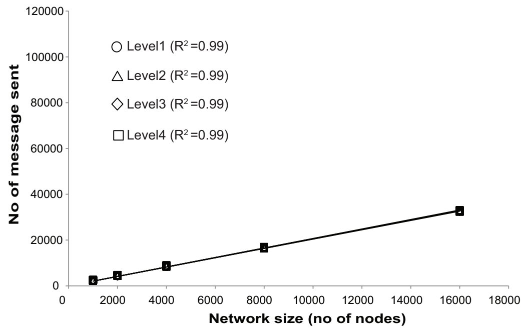
Thus, the total number of simulations scenarios generated was 4 surface levels $\times$ 5 network sizes $\times$ 10 replications $\times$ 2 network configurations = 400. Each of these scenarios was tested against all three algorithms: Takahashi *et al.*'s centralized algorithm, Zhu *et al.*'s decentralized algorithm, and our decentralized DGraF algorithm.

5.2. *Overall scalability*

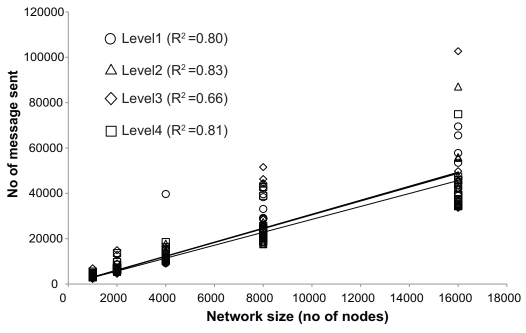
Overall scalability, in terms of communication complexity, was tested by measuring the number of messages generated by the DGraF algorithm and Zhu *et al.*'s algorithm over the five different network sizes (1000, 2000, 4000, 8000, and 16,000 nodes) and four different surface roughness levels with 10 randomized replications at each level. Only the two decentralized algorithms were compared with respect to scalability; Takahashi *et al.*'s algorithm is centralized, and so cannot operate in the network itself.

When comparing DGraF algorithm and Zhu *et al.*'s algorithm, the $|V|$ `ping` initialization messages required by both algorithms were eliminated from the comparison. Further, an additional procedure in Zhu *et al.* (2009) to detect network holes was omitted from our re-implementation. This additional procedure requires additional communication messages, and so must be at best equally scalable to the simplified version we tested, and most likely less scalable.

Figure 5 shows a regression analysis of the results of the experimental runs of the two algorithms over the complete set of irregular networks. As expected from the scalability analysis in Section 4.4, the regression confirmed that our DGraF algorithm was indeed linear in the number of nodes in the network, $O(n)$, generating on average slightly over



a. DGraF algorithm



b.Zhu *et al.* algorithm

Figure 5. Overall scalability for communication messages to identify critical points in a irregular network

two messages per node overall (having discounted the third `ping` message above). Comparing the two algorithms in Figure 5a and b, it can be seen that DGraF algorithm has marginally better efficiency than Zhu *et al.*'s algorithm. A *t*-test revealed that the differences between the two algorithms were significant at the 5% level in most cases (17 out of 20 sets of experimental runs showed a significant reduction in messages sent for the DGraF algorithm).

Further, while our DGraF algorithm shows the a strong linear relationship between network sizes and the number of message sent ($R^2 > 0.99$), there substantial variation in Zhu *et al.*'s algorithm (i.e., $0.66 \leq R^2 \leq 0.83$).

Very similar results were obtained for the regular networks. We conclude that our DGraF algorithm exhibits a small but statistically significant increase in scalability over Zhu *et al.*'s algorithm.

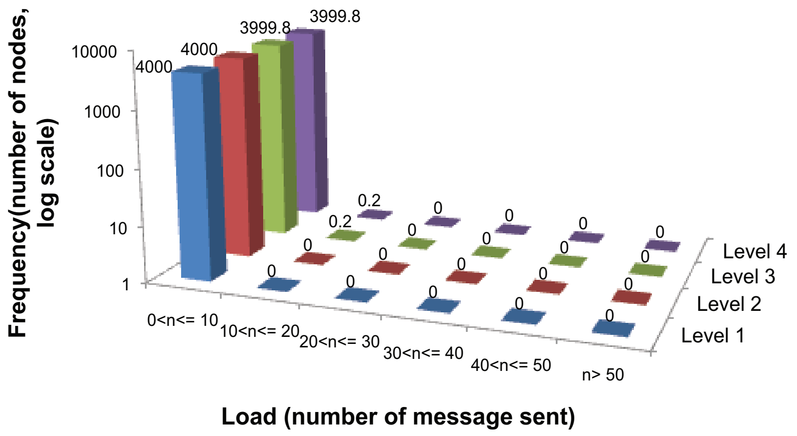
5.3. *Load balance*

In terms of resource constraints in a geosensor network, load balance is frequently more important than overall scalability. Uneven load will lead to depletion of certain nodes' resources, in particular energy, leading to the emergence of holes in network coverage, ultimately with networks becoming disconnected.

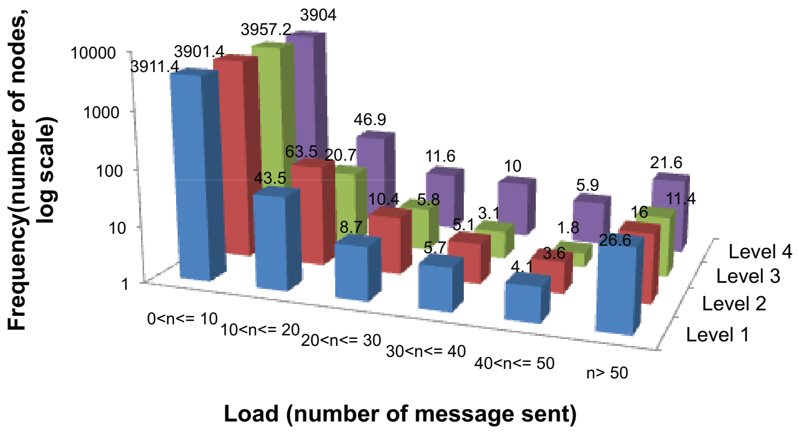
We compared our DGraF' algorithm with Zhu *et al.*'s algorithm, again disregarding the single `ping` message per node required by both algorithms. Each node sends additional messages to identify critical points based on each algorithm's particular properties. In both algorithms, each node must send a minimum of two additional messages. For example, the DGraF algorithm uses `tcel`, `bcel`, and `ctce` messages to identify strong peaks or pits based on ascent or descent vectors, and `pcel` messages to classify a group of passes as one pass. It is expected that unevenly distributed nodes in a network and more rough surfaces will lead to increasingly uneven load.

As an example, a summary of the load distribution in the two decentralized algorithms is illustrated in Figure 6 for the case of irregular networks. Visually comparing these two results clearly indicates an improvement in load balance in the DGraF algorithm. In all cases, our DGraF algorithm resulted in no node sending more than 11 messages; for the Zhu *et al.* algorithm a substantial number of nodes transmitted greater than 50 messages, and in some cases more than 200 messages. Statistical analysis of the full set of results confirms the visual impression.

The algorithm by Zhu *et al.* (2009) also exhibits a slight decrease in load balance with increasing surface roughness, indicated by an increase in the proportion of nodes with more than 10 messages load. This increase was significant at the 95% level between



a. DGraF algorithm



b. Zhu *et al.* algorithm

Figure 6. Load balance for communication messages for irregular networks (averaged over 10 networks of 4000 nodes; roughness level indicated 1–4, smooth to rough).

levels 2, 3, and 4, in the case of irregular networks, and between levels 3 and 4 in the case of regular networks. No significant decreases in load balance were found for the DGraF algorithm across any roughness levels, whether for regular or irregular networks, indicating the relative load stability of the DGraF algorithm.

5.4. *Accuracy*

The section above demonstrated the efficiency of the DGraF when compared with the state of the art decentralized alternative. In this section we also investigate the accuracy of the algorithms, using standard information retrieval measures, positive predictive value (PPV, also frequently termed precision), recall, and F1-score (Buckland and Gey 1994, Goutte and Gaussier 2005). PPV reflects the number of correctly identified critical points as a proportion of all the critical points that were identified by the algorithm. Recall concerns the number of correctly identified critical points as a proportion of all the critical points that should have been identified by the algorithm (i.e., in comparison to the “ground truth”). The F1-score summarizes both the PPV and recall values as the harmonic mean. For example, in this paper a perfect PPV score of 1.0 means that all critical points identified by an algorithm are in reality critical points; a perfect recall score of 1.0 means that all critical points in reality were identified as such by an algorithm.

We compared the PPV, recall, and F1 score of DGraF algorithm, the Takahashi *et al.* algorithm, and the Zhu *et al.* algorithm. The centralized Takahashi *et al.* algorithm used a Delaunay triangulation structure to identify critical points, with knowledge of nodes' coordinate position. The decentralized DGraF and Zhu *et al.* algorithms used a UDG network structure to identify critical points without knowledge of a node's coordinate position. It is therefore expected that the Takahashi *et al.* algorithm will generate the most accurate results. For all algorithms, edge effects were found to lead to false positives at the boundary of geosensor networks. In order to negate edge effects, any identified critical points within a small buffer distance of the network edge were discarded. This procedure was applied to all algorithms consistently, to avoid introducing any bias.

The "ground truth" for this experiment was generated using the centralized Takahashi *et al.* algorithm applied to the original, unsampled surface, containing one million of points (regular grid of 1000 by 1000 points).

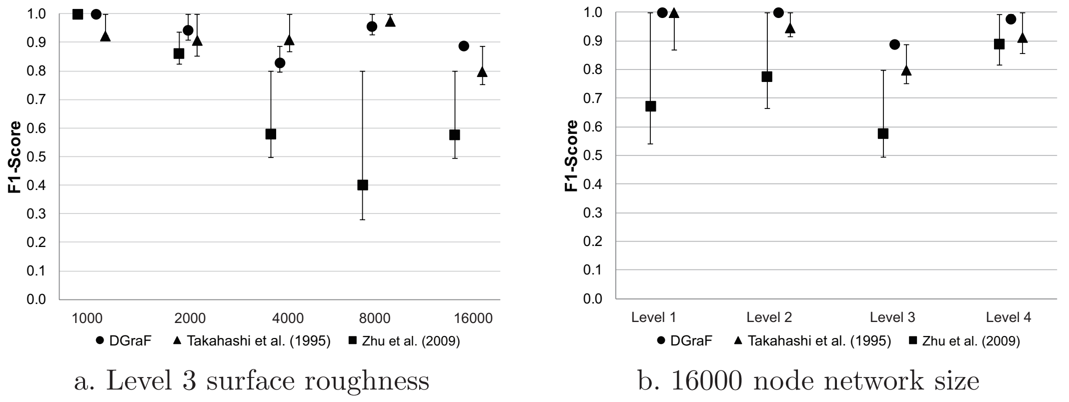


Figure 7. F1-score for peaks in irregular networks for all three algorithms, bounded by bars indicating recall and PPV.

Figure 7 compares typical results for identification of peaks in an irregular network. In Figure 7a, the effect of network size on PPV, recall, and F1-score is shown for a fixed level 3 (moderate) surface roughness. The figure shows that the DGraF performs at a comparable level to Takahashi *et al.*'s algorithm, but tends to outperform Zhu *et al.*'s algorithm especially at larger network sizes. Similarly, Figure 7b shows the effect of changing surface roughness given a fixed network size of 16000 nodes (largest network). In general, again DGraF performs at a comparable level to Takahashi *et al.*'s algorithm, but outperforms Zhu *et al.*'s algorithm especially at smaller network sizes, where spatial granularity effects become more pronounced.

In general, this pattern was found across all the irregular networks and for both peaks and pits. The DGraF algorithm generally performed at a comparable level of accuracy to Takahashi *et al.*, but outperformed the Zhu *et al.* algorithm. Overall, out of the 4 roughness levels and 5 networks sizes tested for peak identification, the DGraF algorithm outperformed (i.e., achieved higher F1-scores than) Takahashi *et al.*'s algorithm in six cases; and was outperformed by (i.e., achieved lower F1-scores than) Takahashi *et al.* in 9 cases. (In the remainder, both algorithms achieved the same performance). In comparison with Zhu *et al.* (2009), the DGraF algorithm performed better in 19 out of 20 cases, performing equally well in the one remaining case. Only in the case of regular networks did Zhu *et al.*'s algorithm achieve marginally higher accuracy: in 17 out of 20 cases DGraF and Zhu *et al.* achieved equal F1-scores; in the remaining three cases Zhu *et al.*'s algorithm outperformed DGraF.

In the case of passes, DGraF and Zhu *et al.* both performed at a similar level of accuracy, but generally below that of Takahashi *et al.*'s algorithm. Takahashi *et al.*'s algorithm outperformed DGraF in 18 out of 20 cases in irregular networks, and 15 out of 20 cases in regular networks. In comparison with Zhu *et al.*'s algorithm, again DGraF performed better in irregular than regular networks. In irregular networks, 14 out of 20 cases DGraF outperformed Zhu *et al.*'s algorithm; in regular networks, 16 out of 20 cases, Zhu *et al.*'s algorithm outperformed DGraF.

5.4.1. Relationship between PPV and recall

The F1-score alone can mask the details of the inverse relationship between PPV and recall. Consequently, further investigation of PPV and recall was conducted. Figure 8a shows the relationship between PPV and recall in a level 4 (most rough) surface, ranging from 4000 to 16,000 node network size. (For space reasons, the two smaller network sizes are omitted.) The PPV-recall curves on the left hand side of the figure show that all three algorithms are able to achieve relatively high levels of both PPV and recall, indicating good performance. It may be observed that in general DGraF algorithm achieved comparable accuracy to Takahashi *et al.*'s algorithm, both of which slightly outperform Zhu *et al.*'s algorithm.

However, it is well known that PPV-recall curves are sensitive to changes in class distribution (Fawcett 2006, Davis and Goadrich 2006). Because the number of true negatives in our experiments greatly outweigh the number of true positives (most nodes are not critical points), the PPV-recall curves in Figure 8 were generated by subsampling negative results (false and true negatives) to achieve comparable class sizes.

A widely used alternative to PPV-recall curves is the receiver operating characteristics (ROC) graph (right hand side of Figure 8). ROC curves are not sensitive to class distribution (Greiner *et al.* 2000, Fawcett 2006), and so enable our entire data set to be analyzed without subsampling. The ROC curves show the trade-offs between false positive rate and true positive rate (i.e., probability that randomly chosen positive instance is ranked above randomly chosen negative one).

The area under the ROC curve (AUC) is a further measure of an algorithm's performance (Bradley 1997). The DGraF's algorithm achieved higher AUC values than Takahashi *et al.* algorithm in 3 out of 5 cases (2000, 4000, 16000 nodes). Moreover, DGraF's algorithm outperformed Zhu *et al.* ' algorithm in all cases. These results are consistent with those of F1-score comparisons. They provide further evidence that DGraF's performance is comparable to the results of Takahashi *et al.* algorithm for the identification of peaks overall.

6. Discussion

Our results demonstrate that the DGraF algorithm is highly scalable, with overall $O(n)$ communication complexity. Although Zhu *et al.*'s algorithm also demonstrated overall $O(n)$ communication complexity, regression analysis for Zhu *et al.*'s algorithm indicated slightly larger constant factors for scalability, and notably lower (R^2) coefficient of determination. We interpret this as evidence that, in a decentralized context, our DGraF algorithm can be relied upon to consume less communication resources overall than Zhu *et al.*'s algorithm.

In terms of load balance, our DGraF algorithm far outperformed Zhu *et al.*'s algorithm, especially in the case of irregular network structures. Further, DGraF load achieved

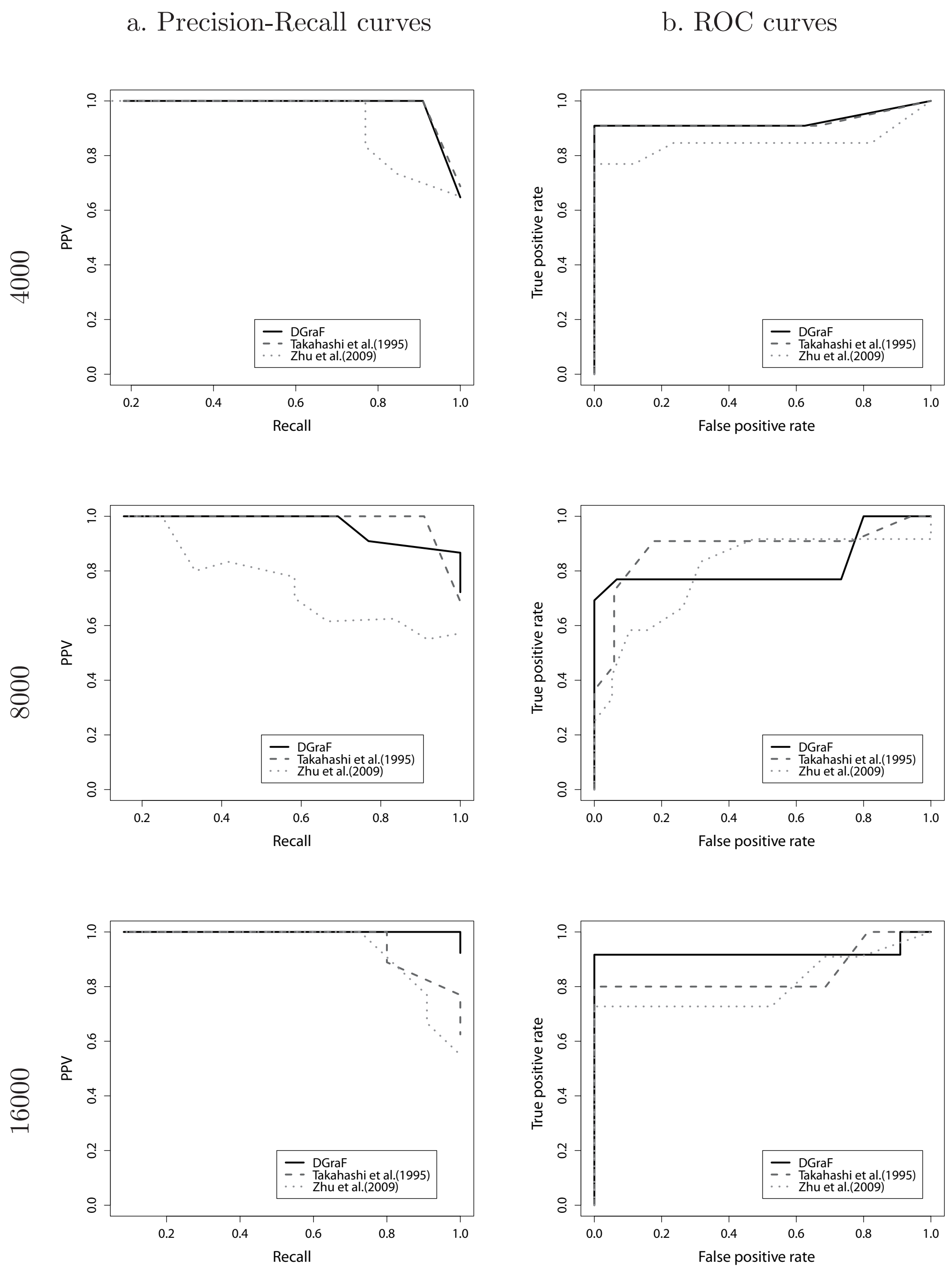


Figure 8. PPV-recall and ROC curves for irregular network sizes 4000–16000 nodes on a level 4 surface for all three algorithms

equally good load balance at all surface roughness levels, where Zhu *et al.*'s algorithm showed some evidence of load balance degradation at higher roughness levels. We attribute this improvement to the use of a "face-based" approach in DGraF. By contrast, Zhu *et al.*'s algorithm must expend substantial additional messages identifying strong peaks and pits, using a threshold, from amongst a number of possibilities.

When it comes to the accuracy of the algorithms, Takahashi *et al.*'s algorithm performed at the highest levels of accuracy overall. This is to be expected as Takahashi *et al.*'s algorithm operated in a centralized mode, rather than in the network, and had access to nodes' coordinate positions. Despite operating decentrally in the network and without any information about nodes' coordinate positions, DGraF achieved comparable (and occasionally better) accuracy for peak and pit identification when compared with Takahashi *et al.*'s algorithm. The DGraF algorithm outperformed Zhu *et al.*'s algorithm in accuracy of peak and pit identification, at least in the case of irregular network structures. In practice, most large geosensor networks are expected to be irregularly structured, due simply to the spatial constraints to network deployment. We attribute the reduced accuracy in Zhu *et al.*'s algorithm to the threshold required to remove noise, and to the sensitivity of the approach to limited granularity and irregular network structures. In general, Zhu *et al.*'s algorithm demonstrated high recall, but at the cost of poor PPV, and so lower F1-scores.

The DGraF algorithm also demonstrated notable robustness to changes in surface roughness and to spatial granularity effects (decreasing network size), maintaining high performance across all these scenarios.

In addition to the identification of peaks and pits, the identification of passes is crucial part to construct the topology of a scalar field. Both decentralized algorithms, DGraF and Zhu *et al.*'s, performed at a lower level than Takahashi *et al.*'s algorithm. Pass identification is inherently a more challenging task for a decentralized than for a centralized algorithm, due to the extended spatial nature of passes, and their dependence on (remote) pits and peaks for their identification. An opportunity for future work exists to improve decentralized pass identification to a level comparable with centralized algorithms, but at limited computational expense.

7. Conclusions

This paper has designed and tested a new algorithm for identifying critical points in a surface network for a monitored spatial field. The algorithm is decentralized, and so can operate in a geosensor network with no centralized control. The approach also requires no coordinate information, only information about each sensor node's neighbors' sensed values. This paper has shown that our decentralized DGraF algorithm can achieve superior efficiency and accuracy when compared with the state-of-the-art decentralized algorithm, and similar accuracy, at least in the case of peak and pit identification, when compared with a commonly used centralized algorithm.

The design of the algorithm was in part in response to the problems of inferring the properties of a continuous surface from its discretization. We show that critical points in a discretization may suggest, but do not imply, critical points in the underlying continuous surface. Thus, our experimental investigation of our algorithm focuses on the empirical relationship between critical points in a continuous surface and its discretization. The results reveal that the limited spatial granularity of a geosensor network dominates the misidentification of critical points. Primarily, large holes in a geosensor network's coverage contribute to the misidentification of critical points in both DGraF and Zhu *et al.*'s algorithm. This discovery provides a focus for future work to combine the detection of holes and the identification of critical points in a network without losing accuracy.

Ultimately, further investigation should focus on the design of a robust algorithm for a time-varying fields, based on the foundations of our DGraF algorithm. The objective is to construct the foundations of efficient and accurate decentralized monitoring of qualitative spatial events in surface networks derived from dynamic scalar fields.

8. Acknowledgments

The authors would like to thank the three anonymous reviewers for insightful and thorough comments, leading to a range of improvements to the paper. Matt Duckham's work is supported by funding from the Australian Research Council (ARC) under the Discovery Projects Scheme, grant numbers DP120103758 and DP120100072. Allison Kealy's work is supported by funding from the ARC, project DP120100072.

References

Akkaya, K., Demirbas, M., and Aygun, R.S., 2008. The impact of data aggregation on the performance of wireless sensor networks. *Wireless Communications and Mobile Computing*, 8 (2), 171–193.

Akyildiz, I., *et al.*, 2002. A survey on sensor networks. *IEEE Communications Magazine*, 40 (8), 102–114.

Alzoubi, K., *et al.*, 2003. Geometric spanners for wireless ad hoc networks. *IEEE Transactions on Parallel and Distributed Systems*, 14 (4), 408–421.

Ayala, R., Fernández, L., and Vilches, J., 2010. The number of critical elements of discrete Morse functions on non-compact surfaces. *Topology and its Applications*, 157 (1), 90–101.

Bajaj, C. and Schikore, D., 1998. Topology preserving data simplification with error bounds. *Computers & Graphics*, 22 (1), 3–12.

Banchoff, T., 1970. Critical points and curvature for embedded polyhedral surfaces. *The American Mathematical Monthly*, 77 (5), 475–485.

Biasotti, S., *et al.*, 2008. Describing shapes by geometrical-topological properties of real functions. *ACM Computing Surveys*, 40 (4), 12:1–12:87.

Biasotti, S., Falcidieno, B., and Spagnuolo, M., 2004. Surface shape understanding based on extended Reeb graphs. *In*: S. Rana, ed. *Topological Data Structures for Surfaces: An Introduction for Geograhpical Information Science*. Willey, 87–102.

Bradley, A.P., 1997. The use of the area under the ROC curve in the evaluation of machine learning algorithms. *Pattern Recognition*, 30 (7), 1145–1159.

Buckland, M. and Gey, F., 1994. The relationship between recall and precision. *Journal of the American Society for Information Science*, 45, 12–19.

Carr, H., Snoeyink, J., and Axen, U., 2003. Computing contour trees in all dimensions. *Computational Geometry: Theory and Applications*, 24, 75–94.

Chiang, Y.J. and Lu, X., 2005. Simple and optimal output-sensitive computation of contour trees. *Computational Geometry: Theory and Applications*, 165–195.

Chong, C.Y. and Kumar, S., 2003. Sensor networks: Evolution, opportunities, and challenges. *Proceedings of the IEEE*, 91 (8), 1247–1256.

Danovaro, E., *et al.*, 2006. A multi-resolution representation for terrain morphology. *In*: M. Raubal, H. Miller, A. Frank and M. Goodchild, eds. *Geographic Information Science.*, Vol. 4197 of *Lecture Notes in Computer Science* Springer, 33–46.

Davis, J. and Goadrich, M., 2006. The relationship between Precision-Recall and ROC curves. *In*: *Proc. 23rd International Conference on Machine Learning*, ICML '06, Pittsburgh, Pennsylvania New York, NY, USA: ACM, 233–240.

Duckham, M., 2013. *Decentralized Spatial Computing: Foundations of Geosensor Networks*. Berlin: Springer.

Edelsbrunner, H., *et al.*, 2004. Time-varying Reeb graphs for continuous space-time data.

In: *Proc 20th Annual Symposium on Computational Geometry*, SCG '04 New York, NY, USA: ACM, 366–372.

Edelsbrunner, H., Harer, J., and Zomorodian, A., 2003. Hierarchical Morse Smale complexes for piecewise linear 2 manifolds. *Discrete and Computational Geometry*, 30, 87–107.

Estrin, D., *et al.*, 2002. Connecting the physical world with pervasive networks. *IEEE Pervasive Computing*, 1 (1), 59–69.

Fawcett, T., 2006. An introduction to ROC analysis. *Pattern Recognition Letters*, 27 (8), 861–874.

Forman, R., 2002. A user's guide to discrete Morse theory. *Sém. Lothar. Combin*, 48, B48c.

Forman, R., 1998. Morse theory for cell complexes. *Advances in Mathematics*, 134 (1), 90–145.

Goutte, C. and Gaussier, E., 2005. A probabilistic interpretation of precision, recall and F-score, with implications for evaluation. *In*: D. Losada and J. Fernández-Luna, eds. *Advances in Information Retrieval.*, Vol. 3408 of *Lecture Notes in Computer Science* Springer, 345–359.

Greiner, M., Pfeiffer, D., and Smith, R., 2000. Principles and practical application of the receiver-operating characteristic analysis for diagnostic tests. *Preventive Veterinary Medicine*, 45 (1), 23–41.

Horanont, T. and Shibasaki, R., 2008. An implementation of mobile sensing for large-scale urban monitoring. *In*: *International Workshop on Urban, Community, and Social Applications of Networked Sensing Systems (UrbanSense08)*.

Kreveld, M.V., *et al.*, 1998. Contour trees and small seed sets for isosurface traversal. *In*: *Proc. 13th Annual ACM Symposium on Computational Geometry*, 212–220.

Krishnamachari, L., Estrin, D., and Wicker, S., 2002. The impact of data aggregation in wireless sensor networks. *In*: *Proc. 22nd International Conference on Distributed Computing Systems Workshops*, 575–578.

Leonard, N., *et al.*, 2007. Collective motion, sensor networks, and ocean sampling. *Proceedings of the IEEE*, 95 (1), 48–74.

Lewiner, T., Lopes, H., and Tavares, G., 2004. Applications of Forman's discrete Morse theory to topology visualization and mesh compression. *IEEE Transactions on Visualization and Computer Graphics*, 10 (5), 499–508.

Lewiner, T., 2012. Critical sets in discrete Morse theories: Relating Forman and piecewise-linear approaches. *Computer Aided Geometric Design*, 1 (1995), 1–13.

Li, X.Y., Calinescu, G., and Wan, P.J., 2002. Distributed construction of a planar spanner and routing for ad hoc wireless networks. *In*: *Proc. 21st Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, Vol. 3 IEEE, 1268–1277.

Lynch, N.A., 1996. *Distributed Algorithms*. San Francisco, CA: Morgan Kaufmann.

Madden, S., *et al.*, 2002. TAG: A tiny aggregation service for ad-hoc sensor networks. *In*: *Proc. 5th Symposium on Operating System Design and Implementation (OSDI)*, 131–146.

MATLAB, 2010. *version 7.10.0 (R2010a)*. Natick, Massachusetts: The MathWorks Inc.

Matsumoto, Y., 2002. *An Introduction to Morse Theory*. Vol. 208. American Mathematical Society.

Maxwell, J.C., 1870. On hills and dales. *Philosophical Magazine Series 4*, 421–427.

Milnor, J., 1969. *Morse Theory*. Princeton University Press.

Murty, R., *et al.*, 2008. CitySense: An urban-scale wireless sensor network and testbed.

In: *IEEE Conference on Technologies for Homeland Security* IEEE, 583–588.

Nittel, S., 2009. A survey of geosensor networks: Advances in dynamic environmental monitoring. *Sensors*, 9 (7), 5664–5678.

Pfaltz, J.L., 1976. Surface Networks. *Surface Networks*, 8, 77–93.

Reininghaus, J., *et al.*, 2010. TADD: A computational framework for data analysis using discrete Morse theory. *Mathematical Software*, 198–208.

Santini, S., Ostermaier, B., and Vitaletti, A., 2008. First experiences using wireless sensor networks for noise pollution monitoring. *In*: *Proc. Workshop on Real-World Wireless Sensor Networks*, REALWSN '08 New York, NY: ACM, 61–65.

Santoro, N., 2007. *Design and Analysis of Distributed Algorithms*. New Jersey: John Wiley & Sons, Inc.

Sarkar, R., *et al.*, 2008. Iso-contour queries and gradient descent with guaranteed delivery in sensor networks. *In*: *Proc. 27th IEEE Conference on Computer Communications (INFOCOM)*, April. Phoenix, AZ: IEEE, 960–967.

Schneider, B. and Wood, J., 2004. Construction of metric surface networks from raster-based DEMs. *In*: S. Rana, ed. *Topological Data Structures for Surfaces: An Introduction for Geographical Information Science*. Willey, 53–70.

Shinagawa, Y., Kunii, T., and Kergosien, Y., 1991. Surface coding based on Morse theory. *IEEE Computer Graphics and Applications*, 11 (5), 66–78.

Skraba, P., *et al.*, 2006. Sweeps over wireless sensor networks. *In*: *Proc. 5th International Conference on Information Processing in Sensor Networks (IPSN)* New York, NY, USA: ACM, 143–151.

Smale, S., 1960. Morse inequalities for a dynamical system. *Bulletin of The American Mathematical Society*, 66 (1), 43–49.

Takahashi, S., *et al.*, 1995. Algorithms for extracting correct critical points and constructing topological graphs from discrete geographical elevation data. *Computer Graphics Forum*, 14 (3), 181–192.

Willensky, U., NetLogo. Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL. Http://ccl.northwestern.edu/netlogo/, 1999. .

Wood, J. and Rana, S., 2000. Constructing weighted surface networks for the representation and analysis of surface topology. *In*: *Proc. Geocomputation*, 23–25.

Zhu, X., Sarkar, R., and Gao, J., 2009. Topological data processing for distributed sensor networks with Morse-Smale decomposition. *In*: *Proc. IEEE Conference on Computer Communications (INFOCOM)* IEEE, 2911–2915.