




# Finding and tracking local communities by approximating derivatives in networks

M. Amin Rigi<sup>1</sup>  · Irene Moser<sup>1</sup> · M. Mehdi Farhangj<sup>2</sup> · Chengfei Lui<sup>1</sup>

Received: 6 December 2016 / Revised: 3 July 2019 / Accepted: 23 September 2019 /

Published online: 2 December 2019

© Springer Science+Business Media, LLC, part of Springer Nature 2019

## Abstract

Since various complex systems are represented by networks, detecting and tracking local communities has become a crucial task nowadays. Local community detection methods are getting much attention because they can address large networks. One famous class of local community detection is to find communities around a seed node. In this research, a novel local community detection method, inspired by geometric active contours, is proposed for finding a community surrounding an initial seed. While most of real world networks are dynamic and the majority of local community detection cannot tackle dynamic networks, the proposed model has the ability to track a local community in a dynamic network. The proposed model introduces and uses the derivative-based concepts curvature and gradient of the boundary of a connected sub-graph in networks. Then, a velocity function based on curvature and gradient is proposed to determine if the boundary of a community should evolve to include a neighbouring candidate. Approximating derivatives in discrete Euclidean space has a long history. However, compared to Euclidean space, graphs follow a non-uniform space in which the dimensionality, given by the fluctuation in degrees of nodes, fluctuates from one node to another. This complexity complicates the approximation of derivatives which are needed for defining the curvature and gradient of a node in the boundary of a community. A new framework to approximate derivatives in graphs is proposed for such a purpose. For finding local communities, benchmarking our method against two recent methods indicates that it is capable of finding communities with equal or better conductance; and, for tracking dynamic local communities, benchmarking of the proposed method against ground-truth dataset shows a noticeable level of accuracy.

**Keywords** Derivative · Graph · Dynamic · Differential geometry · Local community detection · Active contours

---

✉ M. Amin Rigi  
mrigi@swin.edu.au

## 1 Introduction

Since communities are at the heart of human society, community analysis has always been one of the chief topics in sociology. Although there is no unique definition of a community amongst researchers [27], a community, cluster, or module can be seen as a set of nodes which share similar properties such as features, objectives, and roles. With information technology revolutionising almost every aspect of life in the last decade, social networks have become immensely popular both in industry and academia.

Depending on the availability of network information, community detection methods are categorised into global and local approaches. Global community detection algorithms need global information of the entire network and are computationally expensive when applied to all but the smallest networks. Considering these limitations, it is preferable to address the problem of community detection from a local perspective. Local methods try to find community patterns in subsets of a graph without considering the global information. This avoids the need to calculate and consider information about the entire graph whenever a node is included in a community. Also, in some cases such as the World Wide Web or the Internet the information of the entire network is not available. For these reasons, local community detection methods scale more easily to large networks of interest.

As a result, many researchers try to analyse communities by focusing on a particular region of the network. In most local community detection approaches, the goal is to find the communities around a seed node. Web search engines are interested to find similar pages around a particular page. HITS [30] and PageRank [58] are some of popular ranking algorithms which can be seen as local community detection algorithms in the network of the Web.

Communities can also be investigated based on the dynamics of the networks. As most real-world networks change over time, analysing and tracking dynamic communities is a crucial task. Surprisingly, the majority of studies in this area consider communities in static networks.

In order to address some shortcomings of the existing methods, especially efficiency and the ability to trace dynamic communities, we propose the Derivative-based Community Detection (DCD) method inspired by geometric active contours [7]. Geometric Active Contours are used extensively for detecting objects in 2D images in the field of machine vision. They are known for their speed, their autonomous and unsupervised nature, and their ability to track dynamic objects. The analogy between the discovery of shapes in images and the detection of communities in graphs suggests that an application of the active contours method to graph spaces might provide an efficient alternative to existing community detection techniques.

In geometric active contours, the boundary of an object evolves until it accurately reflects the edge of an object, which is defined by large differences in the pixel values. The boundary of the object is defined in terms of curvature and gradient, which are based on derivatives. Curvature and gradient constitute the velocity of the boundary, which determines whether the boundary moves to include further pixels.

This principle of moving a boundary until all suitable nodes have been included can be translated into graph space provided we can determine the derivatives of a function in a graph. In this work, which is the extended version of our previous work [47], we provide a novel means of approximating derivatives of functions in graph space and use these derivatives to find the boundaries of communities and, also, to update the boundaries when network changes in dynamic settings. The contributions of this research are providing a general framework with feasible time complexity for approximating derivatives in graph

space; proposing a derivative based method for detecting local communities; providing an effortless method using the proposed local community detection approach to track local communities in dynamic networks; and, addressing and covering the gap between image processing and networks. In the experiments, we compare our proposed local community detection method with two other local community detection methods presented in the work of Jeub et al. [27].

The rest of this paper is organized as follow: Section 2 reviews related works, Section 3 presents active contours and the application of differential geometry in image processing, Section 4 generalizes the differential geometry to graph space, and finally in Sections 5 and 6 the proposed derivative approach has been validated on community detection and tracking applications.

## 2 Related work

Community detection is a well studied area [33, 38]. Detecting communities is an important problem in various domains. In sociology, in order to discover human behaviour patterns, researchers have studied social communities for decades [14, 31]. In biology and cancer research, researchers are looking for communities in protein interaction networks which lead to some specific actions in the cell [10, 46]. Community detection algorithms have been extensively used in computer science. For example, clustering Web clients in the World Wide Web is used to improve the performance of service providers [32], or finding communities in social networks. Community detection approaches can be classified by the amount of information available about the graph. If the information of the entire network is available, the method is called a global method; otherwise, it is called local method. In this section, firstly, global and local methods are briefly reviewed. Secondly, the problem of tracking communities in dynamic networks is reviewed. Thirdly, we investigate the problem of finding a local community around a seed node. Finally, since one main contribution of this research is a framework for approximating derivatives in graph space, the studies in this domain are reviewed.

### 2.1 Global community detection

In global community detection algorithms, the data on the entire network is available. In this class of algorithm each method has a quality function which takes the partitions as input and returns a number as a quality measure. Since there is no global census regarding the definition of a community, different methods consider different criteria for defining quality functions. The main drawback of global community detection is that it has to extract pairwise information for all pairs of nodes in the entire graph.

Even though most of the complex networks in the real world are dynamic, the larger part of the studies focuses on static global community detection methods. Numerous survey papers on the topic exist [16, 22, 49, 50].

Spectral Clustering uses the eigenvectors of Laplacian adjacency matrices of graphs, and then eigenvectors are used as a basis of a clustering algorithm such as hierarchical or K-means in order to clusters vertices into communities [41, 45]. Modularity [13, 18] is a well-known quality function in community detection research. Modularity based methods assume that the edges that have the highest degree of betweenness centrality connect two communities. The algorithm finds the betweenness centrality for every edge and removes the one with highest degree at each step until graph becomes disconnected, and splits into

two partitions. The same procedure is repeated in each partition. Modularity optimisation is proved to be an NP-hard problem [5]. Correlation clustering assumes edges have real-valued weights. It tries to maximise the sum of the intra-cluster edge weights, or, alternatively, minimise the sum of the extra-cluster weights [9]. In both cases, finding the optimal solution is an NP-hard problem [4].

## 2.2 Local community detection

In local community detection, researchers assume that communities have independent patterns. They try to find community patterns in subsets of a graph without considering global information. They only need to calculate information for a part of a graph. Therefore, they are more applicable to real-world large networks such as the Internet.

In local community detection, most algorithms try to find a community that surrounds a node or a seed. Many algorithms in this category are extensions of global community detection algorithms. In local modularity, one defines a quality function for one community, and then, in an agglomerative procedure, adds nodes to the community [12, 34]. At each step of this class of algorithms a candidate node which has the highest quality (based on local modularity) is added to the community until the maximum size of the community has been reached. Mahoney et al. [39] proposed the local spectral clustering algorithm discussed before. Anderson and Lang [2] used random walks in order to find local communities. When random walks start with a small number of steps from an initial seed node, the random walks are more likely to become trapped in the same community rather than traveling to other communities. Another problem in local community detection in social networks is how to find different social circles of a node. A node may belong to more than one community, which induces overlapping communities. The majority of studies in this domain are categorised into two domains: Detecting communities based on network structure [1, 43] and based on nodes features [28]. There are very few works that consider both structure and attributes [40, 62]. In this research, we propose a method which is able to find overlapping and hierarchical communities of a user (or seed) by only considering the structure of network. The proposed local community detection method, DCD, belongs to seed-expansion methods. Seed-expansion methods are a class of algorithms that find local communities around a seed node [3, 13, 39].

## 2.3 Tracking communities in dynamic networks

Based on the nature of changes (over time) or the nature of the algorithms, there are different categorisation for tracking communities.

Two approaches based on changes are snapshot and temporal smoothness. In the snapshot model, different snapshots of the network at different time steps are available and the task is to use find communities or clusters and track clusters over time in order to interpret their evolution over time [61]. In the temporal smoothness model, the goal is to derive the communities over time given a stream of atomic changes.

Two approaches based on the nature of tracking algorithms are evolutionary clustering and dynamic clustering. In evolutionary clustering, after each change, communities are discovered using a static clustering method whereas in dynamic clustering the goal is to update the clusters in the next time step using the information of the current time step.

### 2.3.1 Snapshot model

In the snapshot model, snapshots of the evolving network at different time steps are available. Due to emergence or disappearance of nodes or edges, there exist a collection of changes between snapshots. In the first snapshot at  $t = 0$  there are  $n$  communities  $\{C_1, \dots, C_n\}$ . In a more general case,  $n$ , the number of communities, can also change over time. This means communities can be born, die, or merged. Each node belongs to a certain community. The job of community maintenance algorithms is to update communities (membership of nodes to communities) in the next time steps ( $t = 1, 2, \dots$ ), and then to analyse the evolution of communities over time.

Since in the snapshot model at each time step the fate of very node should be decided, the algorithms in this domain should consider the entire network. As a result, the community tracking models in this framework are usually adaptations of global community detection methods.

### 2.3.2 Temporal smoothness

In temporal smoothness, the available information is the graph at  $t = 0$ , and a continuous stream of atomic changes. An atomic change can be node/edge addition deletion. Up to now, all studies in temporal smoothness fall into global community detection category. That is to say they assume at  $t = 0$ , the initial network is partitioned into  $n$  clusters  $\{C_1, \dots, C_n\}$ . Each node belongs to a certain community. Then, with the stream of atomic changes, the cluster and membership for each node a cluster is updated.

### 2.3.3 Tracking global communities in dynamic networks

In snapshot model, using evolutionary methods, one takes different snapshots of network, finds communities in each snapshot with a static clustering model, and, then, interprets their change over time [61]. Basically, all methods in Section 2.1 can be used in such settings.

In temporal-smoothness, the goal is to derive the communities over time given a stream of changes. A change can be the addition or removal of a node or edge. Falkowski et al. [19] use Girvan-Newman modularity-based community detection for both finding and tracking communities. Tong et al. [57] suggested low rank approximation for detecting dynamic networks; however, their research lacks evaluation. Xu et al. [60] used a hidden Markov model to address dynamic networks. In Verted-centred methods [6, 29], which have similar concept as K-means clustering algorithm, evolving leaders and, therefore, the communities around leaders are found in each time step. Leskovec et al. [36] used the Clique Percolation Method (CPM) to identify communities at each time step, and then match them with community evolution methods. MONIC, a framework for modeling and monitoring clusters transitions over time, was suggested by Spiliopoulou [53]. Graphscope [54] is a parameter-free algorithm which mines time evolving graphs in order to find communities, and their change over time. Nguyen et al. [42] developed a framework for identifying and tracking overlapping communities by defining a global objective function which is summation of a set of local communities. Samie et al. [48] developed a two-phased model that is comprised from a global and local method. In the first phase, they find global communities and, in the second phase, the find and track local communities in the detected clusters using the global

approach. Shang et al. [51] proposed a learning based approach for tracking global communities in dynamic networks. They train and use a classifier in order to find and inspect the vertices that are more likely to change their community after the network is changed.

## 2.4 Tracking local communities in dynamic networks

Based on the categorisation of previous section, local communities can be investigated in four different ways.

1. **Snapshot model:** There are several changes in each time step.
  - **Evolutionary clustering:** Any local community detection method from Section 2.2 can simply be used at each time step.
  - **Dynamic clustering:** In this category, a method must be able to maintain and update the local community given several changes, and the current structure of the community. With the assumption that the changes between two snapshots are not considerable, only few studies [55, 56] have tackled this problem. Takaffoli et al. [56] use static L-metric method [11] to find local communities in each snapshot from using the structure of the local communities in the previous snapshot.
2. **Temporal smoothness:** Stream of atomic changes.
  - **Evolutionary clustering:** Local methods can be used after each atomic change. For example, Hu et al. [26] proposed a personalised page rank algorithm that finds the local communities in the area of changes after each change atomic change.. This class of algorithms might not be computationally efficient because after every single atomic change, local community detection method should run.
  - **Dynamic clustering:** The goal in this category is to update the structure of a local community after every atomic change without recalculations. According to best of our knowledge, there is no other methods under this setting. DCD belongs to this category.

## 2.5 Derivatives in graphs

Friedmann and Tillich [23] developed a connection between graph theory and calculus. The authors defined continuous concepts like continuously differentiable functions, boundary, and gradient over the graph space to develop a wave equation in order to understand the changes in the connectivity of graph nodes. In this domain, Diao et al. [17], explored a bounded symmetric function defined over the edges of a finite labeled graph called graphon space. They proposed a general theory of differentiation over this space. As this space is not a vector space, the authors refined Gateaux derivative to make it appropriate for graphon space. Both of these studies proposed partial differential equations (PDEs) over a continuous topology given on a graph. In an attempt to avoid complex differential theory and to take advantage of finite dimensional linear algebra, an alternative approach is to formulate derivatives on the original discrete graph space.

However, according to Solomon [52], the derivatives cannot be derived as easily as in the continuous domain; Solomon modeled the continuous Laplacian in the discrete domain of graphs by finite difference approximation. Inspired by discretisation of continuous PDEs, the Laplacian was computed for each two nodes in the graph and used to solve Laplace and Poisson equations in the graphs domain. Solomon approach has its own limitations which makes it impractical and difficult to use. Solomon [52] mentions: “*Differential graph*

*analysis, on the other hand, requires the development of a new framework to understand diffusion, oscillation, and other phenomena as they might occur on graphs, which may not be physically realizable.”.*

In this research, it is shown that extracting differential geometry properties using the proposed framework is fairly easy – with some intuitive imagination. Moreover, when it comes to finding higher order derivatives (second or higher) Solomon’s framework is computationally unfeasible since it needs to solve a challenging combinatorial problem whereas the time complexity of the proposed framework is polynomial. The proposed framework finds the derivatives by solving systems of linear equations which is considerably faster than Solomon’s approach [52]. The proposed approach also does not deal with the mathematical difficulty and limitations of Friedmann and Tillich [23] and Diao et al. [17] approaches.

### 3 Geometric active contours

In the field of image processing, the problem of object detection has been addressed in many different ways. Active Contours is a method devised first in 1988 [7]. A related approach, based on differential geometry, was devised in 1997. Due to its efficiency, autonomy, and unsupervised nature geometric active contours [7, 20, 21] are used extensively for detecting objects in 2D images in the field of machine vision. In this method, an initial contour deforms and evolves in order to find the boundary of an object. In an image, shapes distinguish themselves from the background by boundaries characterised by pixels whose properties are very different from those of the adjacent pixels which form part of the background.

Initially, a curve is created at a random location of the image with the goal of finding the boundary of an object. The curve evolves based on two concepts, curvature and gradient. The curvature of a function  $f(x)$ , defined in (1), describes how fast the curve changes its tangent or direction.

$$\kappa = \frac{f''(x)}{(1 + f'^2(x))^{\frac{3}{2}}} \quad (1)$$

$f'(x)$  and  $f''(x)$  are the first and second derivatives. The vector differential operator  $\nabla$  has the following definition:

$$\nabla = \frac{\partial}{\partial x}i + \frac{\partial}{\partial y}j + \frac{\partial}{\partial z}k \quad (2)$$

Assuming three-dimensional Euclidian space, the gradient of  $f(x, y, z)$  is obtained by applying the vector operator  $\nabla$  to the scalar function  $f(x, y, z)$  as defined in (3).

$$\nabla f(x, y, z) = \frac{\partial f}{\partial x}i + \frac{\partial f}{\partial y}j + \frac{\partial f}{\partial z}k \quad (3)$$

In geometric active contours, the curve evolves in the direction that is perpendicular to the curve. The curve is considered the current boundary, and an adjacent pixel on the movement direction of the boundary is evaluated for inclusion based on the magnitude of the gradient between the pixel on the boundary and the neighbouring pixel in that direction. In images, the gradient is obtained by subtracting the grey values of neighbouring pixels. A second criterion for the inclusion of a neighbouring pixel is the curvature of the current boundary. A straight line has a curvature of zero. A curve that ‘recedes’ inward towards the shape has a high curvature. Intuitively, an object is likely to strive to include ‘inserts’

into its area. Hence an increased curvature favours the inclusion of pixels on the outside of the boundary. In combination, gradient and curvature result in the velocity  $s$  of the curve, expressed in (4). The velocity decides the likelihood of a pixel being included in the shape.

$$s = g\kappa\mathbf{n} - (\nabla g\mathbf{n})\mathbf{n}, \text{ where } g = \frac{1}{1 + |\nabla I|^2} \tag{4}$$

where  $\mathbf{n}$  denotes the normal direction and  $I$  the pixel values in an image with  $|\nabla I|$  as the magnitude of the gradient between two pixels [7].

### 4 Differential geometry in graph space

Image processing is a data-intensive process which benefits from localised methods like active contours. Graphs as encountered in social networks are similarly demanding because of the potential sizes of graphs and their high dimensionality. The analogy between the discovery of shapes in images and the detection of communities in graphs suggests that an application of the active contours method to graph spaces might provide an efficient alternative to existing clustering techniques.

Mapping the relevant concepts from Euclidean to graph space poses a few challenges. While in image processing, the goal is to identify shapes with an external boundary, communities in graphs are defined as sets of nodes that share more properties with other nodes within the same community than they do with nodes outside the community. Unlike images, where the number of dimensions is uniform across the pixels, each node in a graph can have different numbers of neighbours, giving rise to high fluctuations in dimensionality. An image has a clearly defined boundary, whereas it is hard even to define the boundary of an entire graph. As a consequence of the non-uniform dimensions of a graph, most matrix operations used in machine vision cannot be applied to graphs.

In order to be able to identify communities using geometric features, it is necessary to define shapes and their boundaries in a graph. We define a shape  $\chi(v, \xi)$  in a graph  $G(V, E)$  as a connected subgraph of a set of nodes  $v$  that are connected by the edges  $\xi$ ; ( $v \subseteq V, \xi \subseteq E$ ). The boundary of a shape in a graph is defined as the nodes inside the shape that have common edges with nodes outside the shape, formally a node  $v_i$  is on the boundary of shape  $\chi$  if  $\exists e_{ij} \in E | v_i \in v \wedge v_j \in V \wedge v_j \notin v$

These connected nodes in the graph space create an object with its boundaries. The following figure demonstrates two shapes in the two different graphs.

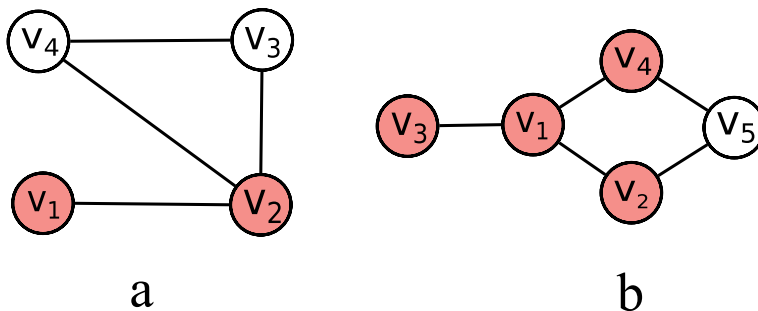


Figure 1 Examples of shapes and their boundaries in graphs



In Figure 1a, the nodes in red colour compose a shape which consists of only two nodes. Figure 1b shows a shape that is comprised of 4 nodes. Nodes  $v_2$  and  $v_4$  in Figure 1b are considered the external boundary of the shape. Having defined the concept of shapes and boundaries in graph space, we have to answer the question how a boundary can move to align with desirable communities in graphs. To evolve boundaries, a method requires criteria which decide whether a node in the neighbourhood of the boundary is to be included in the shape. This can be achieved by applying the concept of velocity (4) to graph spaces. Velocity is calculated using the components of gradient and curvature, both of which require the calculations of derivatives.

### 4.1 Derivatives in graphs

The notion of velocity in image processing is based on derivatives which produce gradient and curvature on the boundary of a shape. In a graph, we define the derivative as the rate of change of a function  $F(v_i)$  for a given node  $v_i$ . Function  $F(v_i, v_p)$  can be defined as the similarity, match or the difference between node  $v_i$  and a constant node  $v_p$ . For example,  $F_{degree}(v_i, t)$  can represent the degree of the node  $v_i$  at time  $t$  in a dynamic network, or  $F(v_i, v_p, t)$  can represent the similarity of the node  $v_i$  to the node  $v_p$  at the time  $t$ .

Since derivatives do not exist in discrete space, the Taylor series (5) expansion can be used to approximate derivatives.

$$f(x + \Delta x) = f(x) + (\Delta x)f'(x) + \dots + \frac{(\Delta x)^i}{i!} f^i(x) + \dots \tag{5}$$

In the remainder of this section we will show how the derivatives in graphs can be approximated from the Taylor series. Calculating the curvature of a function requires the first and second derivatives.

According to the Taylor series, the numerical approximation of the first-order derivatives for a function  $f(x)$  is [15]:

$$\frac{\partial f}{\partial x} = \frac{f(x + \Delta x) - f(x)}{\Delta x} + O(\Delta x) \tag{6}$$

Similarly, the first-order backward derivative is:

$$\frac{\partial f}{\partial x} = \frac{f(x) - f(x - \Delta x)}{\Delta x} \tag{7}$$

The second-order first derivative is:

$$\frac{\partial f}{\partial x} = \frac{f(x + \Delta x) - 2f(x) + f(x - \Delta x)}{2\Delta x} + O(\Delta x)^2 \tag{8}$$

The terms  $O(\Delta x)$  and  $O(\Delta x)^2$  in (6) and (8) are called truncation error and represent the remaining parts on the right side of the (5) which are neglected if one wishes to approximate derivatives.

The approximations in (6) and (8) can be extended to graphs. As an example, we will determine the derivative of  $F(v_i)$  in Figure 2.

First-order derivative:

$$F'_v(v_i) \approx \frac{F(v_{i+1}) - F(v_i)}{v_{i+1} - v_i} \tag{9}$$

$v_{i+1} - v_i$  shows the distance, or dissimilarity between these two nodes (e.g. difference of two friends in a social network). Any derivative based operation needs a distance function. Since this paper only focuses on the structures of networks for finding communities,

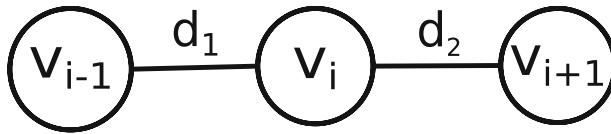


Figure 2 Example graph with three nodes

structural distance or similarity measures in networks are considered. Nodes are structurally equivalent if they are in the same area of the graph and have the same neighbours. Three main structural similarity measures are simple structural equivalence, Cosine similarity measure, and Pearson product-moment correlation coefficient. In this research, normal structural distance (45) is used. Analysing different distance measures and their effect on detecting communities are left for future works.

Second-order first derivative:

$$F'_v(v_i) \approx \frac{F(v_{i+1}) - 2F(v_i) + F(v_{i-1}))}{d_1 + d_2} \tag{10}$$

where  $d_1 = v_i - v_{i-1}$  and  $d_2 = v_{i+1} - v_i$ .  $d_i$ , in general, shows the difference between the nodes in the graph. Applying the model to weighted graphs is straightforward: If the weight between  $v_i$  and  $v_{i-1}$  is  $w$ , then  $d_i(w) = \frac{d_i}{w}$ .

The second derivative according to the Taylor series:

$$f(x + \Delta x) = f(x) + \Delta x f'(x) + \frac{(\Delta x)^2}{2!} f''(x) + O(\Delta x)^3 \tag{11}$$

In the rest of this section, after analysing two examples, a general model for finding the derivatives for a given node is proposed.

Example 1 Approximating derivatives for a node with two neighbours (Figure 3):

$$F(v_c + d_1) = F(v_1) = F(v_c) + d_1 F'_v(v_c) + \frac{d_1^2}{2} F''_v(v_c) \tag{12}$$

$$F(v_c + d_2) = F(v_2) = F(v_c) + d_2 F'_v(v_c) + \frac{d_2^2}{2} F''_v(v_c) \tag{13}$$

This can be shown and solved as a linear system with two equations and two unknowns:

$$\begin{bmatrix} d_1 & \frac{d_1^2}{2} \\ d_2 & \frac{d_2^2}{2} \end{bmatrix} \begin{bmatrix} F'(C) \\ F''(C) \end{bmatrix} = \begin{bmatrix} F(V_1) - F(C) \\ F(V_2) - F(C) \end{bmatrix} \tag{14}$$

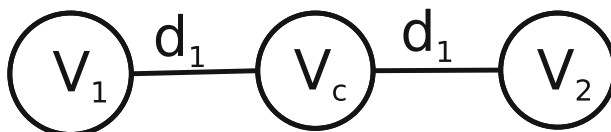
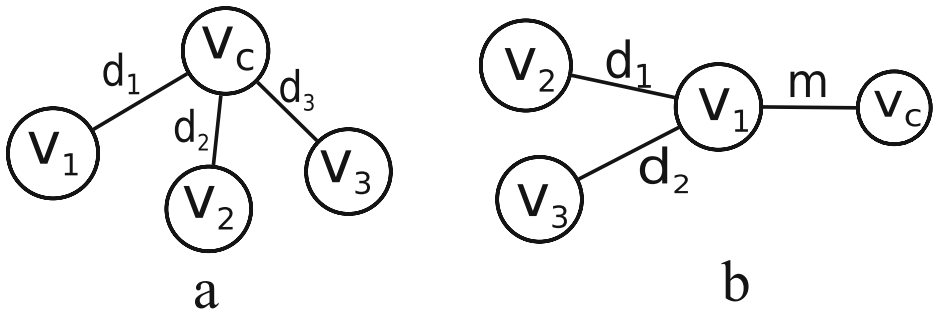


Figure 3 Node with two neighbours



**Figure 4** a.  $v_c$  a node with three neighbours; b.  $v_c$  a non-central node

*Example 2* Approximating the derivatives of  $F(v_c)$  in Figure 4a:

$$F(v_c + d_1) = F(v_1) = F(v_c) + d_1 F'_v(v_c) + \frac{d_1^2}{2} F''_v(v_c) \tag{15}$$

$$F(v_c + d_2) = F(v_2) = F(v_c) + d_2 F'_v(v_c) + \frac{d_2^2}{2} F''_v(v_c) \tag{16}$$

$$F(v_c + d_3) = F(v_3) = F(v_c) + d_3 F'_v(v_c) + \frac{d_3^2}{2} F''_v(v_c) \tag{17}$$

Accordingly, the system of equations is:

$$\begin{bmatrix} d_1 & \frac{d_1^2}{2} \\ d_2 & \frac{d_2^2}{2} \\ d_3 & \frac{d_3^2}{2} \end{bmatrix} \begin{bmatrix} F'(v_c) \\ F''(v_c) \end{bmatrix} = \begin{bmatrix} F(v_1) - F(v_c) \\ F(v_2) - F(v_c) \\ F(v_3) - F(v_c) \end{bmatrix} \tag{18}$$

This is an overdetermined system with three equations and two unknowns. Overdetermined systems are usually inconsistent with no unique solution. In this case, one way of solving the problem of overdetermination is to convert an overdetermined system to a determined one by adding more unknowns in the form of higher derivatives, of course at the cost of additional complexity. Higher order derivatives are impractical in the case of nodes that have a larger number of neighbours. Other alternatives are the least squares approximation methods discussed later.

Here, the unknowns are higher derivatives. Although higher orders of derivatives are not necessary in the curve evolution of the geodesic active contour technique, at the price of higher computations, we will achieve higher accuracy for the first and second derivatives by considering them.

$$F(C + d_1) = F(V_1) = F(C) + d_1 F'_V(C) + \frac{d_1^2}{2} F''_V(C) + \frac{d_1^3}{3!} F'''_V(C) \tag{19}$$

$$F(C + d_2) = F(V_2) = F(C) + d_2 F'_V(C) + \frac{d_2^2}{2} F''_V(C) + \frac{d_2^3}{3!} F'''_V(C) \tag{20}$$

$$F(C + d_3) = F(V_3) = F(C) + d_3 F'_V(C) + \frac{d_3^2}{2} F''_V(C) + \frac{d_3^3}{3!} F'''_V(C) \tag{21}$$

$$\begin{bmatrix} d_1 & \frac{d_1^2}{2} & \frac{d_1^3}{3!} \\ d_2 & \frac{d_2^2}{2} & \frac{d_2^3}{3!} \\ d_3 & \frac{d_3^2}{2} & \frac{d_3^3}{3!} \end{bmatrix} \begin{bmatrix} F'(C) \\ F''(C) \\ F'''(C) \end{bmatrix} = \begin{bmatrix} F(V_1) - F(C) \\ F(V_2) - F(C) \\ F(V_3) - F(C) \end{bmatrix} \tag{22}$$

*Example 3* Approximating the derivatives of  $F(v_c)$  for the non-central node  $v_c$  in Figure 4b. After extracting the equations based on the Taylor series, similar to example 1, the system of the equations is (Fig. 5):

$$F(v_c + m) = F(v_1) = F(v_c) + mF'_v(v_c) + \frac{m^2}{2}F''_v(v_c) + \frac{m^3}{3!}F'''_v(v_c) \tag{23}$$

$$F(v_c + m + d_1) = F(v_2) = F(v_c) + (m + d_1)F'_v(v_c) + \frac{(m + d_1)^2}{2}F''_v(v_c) + \frac{(m + d_1)^3}{3!}F'''_v(v_c) \tag{24}$$

$$F(v_c + m + d_2) = F(v_3) = F(v_c) + (m + d_2)F'_v(v_c) + \frac{(m + d_2)^2}{2}F''_v(v_c) + \frac{(m + d_2)^3}{3!}F'''_v(v_c) \tag{25}$$

$$\begin{bmatrix} m & \frac{m^2}{2} & \frac{m^3}{3!} \\ (m + d_1) & \frac{(m + d_1)^2}{2} & \frac{(m + d_1)^3}{3!} \\ (m + d_2) & \frac{(m + d_2)^2}{2} & \frac{(m + d_2)^3}{3!} \end{bmatrix} \begin{bmatrix} F'(v_c) \\ F''(v_c) \\ F'''(v_c) \end{bmatrix} = \begin{bmatrix} F(v_1) - F(v_c) \\ F(v_2) - F(v_c) \\ F(v_3) - F(v_c) \end{bmatrix}$$

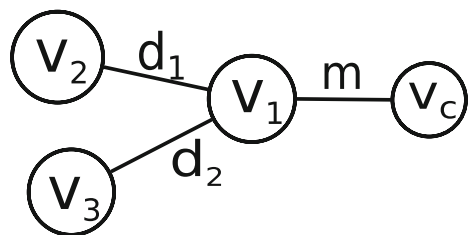
begincenter .

$$\begin{bmatrix} m & \frac{m^2}{2} & \frac{m^3}{3!} \\ (m + d_1) & \frac{(m + d_1)^2}{2} & \frac{(m + d_1)^3}{3!} \\ (m + d_2) & \frac{(m + d_2)^2}{2} & \frac{(m + d_2)^3}{3!} \end{bmatrix} \begin{bmatrix} F'(v_c) \\ F''(v_c) \\ F'''(v_c) \end{bmatrix} = \begin{bmatrix} F(v_1) - F(v_c) \\ F(v_2) - F(v_c) \\ F(v_3) - F(v_c) \end{bmatrix}$$

**4.1.1 A general framework for approximating derivatives of a function in the graphs**

The approximation of derivatives of nodes depends on the position of the node. When a node has a degree of one, we consider it a non-central node. Nodes with degrees larger than one are central nodes. For the purpose of approximating derivatives, if a node has only one

**Figure 5** Non-central nodes



neighbour, it is only possible to calculate the first derivative. Unlike with central nodes, in the case of non-central nodes it is necessary to consider neighbours of neighbours to obtain higher-order derivatives.

**Central nodes (Figure 6a)** A central node  $v_c$  has  $n$  neighbours. The Taylor series equation for the  $i$ -th ( $1 \leq i \leq n$ ) neighbouring node of  $v_c$  can be written as:

$$F(v_c + d_i) = F(v_i) = F(v_c) + d_i F'_v(v_c) + \dots + \frac{d_i^n}{n!} F^n_v(v_c) \tag{26}$$

The equations system is:

$$\begin{bmatrix} d_1 & \dots & \frac{d_1^n}{n!} \\ \dots & \dots & \dots \\ d_n & \dots & \frac{d_n^n}{n!} \end{bmatrix} \begin{bmatrix} F^1(v_c) \\ \dots \\ F^n(v_c) \end{bmatrix} = \begin{bmatrix} F(v_1) - F(v_c) \\ \dots \\ F(v_n) - F(v_c) \end{bmatrix} \tag{27}$$

It is a system of linear equations with  $n$  equations and  $n$  unknowns.

**Non-central nodes** Figure 6b shows one of the peripheral nodes as the node  $v_c$  for which we approximate the derivatives. The neighbour of  $v_c$  is always a central node unless it is part of a two-node component of the graph, in which case it is only possible to calculate the first derivative.

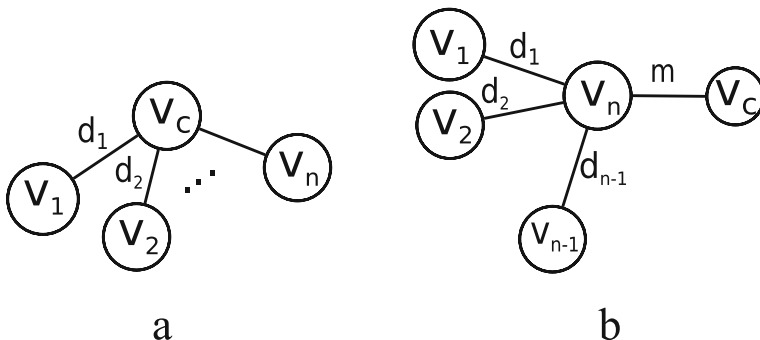
In this case,  $v_n$  has several neighbours, therefore the derivatives of  $F(v_c)$  can be approximated by solving the following system of equations:

$$F(v_c + m) = F(v_n) = F(v_c) + m F'_v(v_c) + \dots + \frac{m^n}{n!} F^n_v(v_c) \tag{28}$$

For all other nodes  $V_i$  where  $1 \leq i \leq n - 1$  we have the following equation:

$$\begin{aligned} F(v_c + m + d_i) &= F(v_i) = \\ F(v_c) + (m + d_i) F'_v(v_c) + \dots + \frac{(m+d_i)^n}{n!} F^n_v(v_c) \end{aligned} \tag{29}$$

$$\begin{bmatrix} m & \dots & \frac{m^n}{n!} \\ (m + d_1) & \dots & \frac{(m + d_1)^n}{n!} \\ \dots & \dots & \dots \\ (m + d_{n-1}) & \dots & \frac{(m + d_{n-1})^n}{n!} \end{bmatrix} \begin{bmatrix} F'(v_c) \\ F''(v_c) \\ \dots \\ F^n(v_c) \end{bmatrix} = \begin{bmatrix} F(v_n) - F(v_c) \\ F(v_1) - F(v_c) \\ \dots \\ F(v_{n-1}) - F(v_c) \end{bmatrix}$$



**Figure 6** a. Node  $v_c$  with  $n$  neighbours; b. Node  $v_c$  with one neighbour

### 4.1.2 Derivatives in inconsistent and overdetermined systems

In general terms of linear algebra, a system of equations is expressed as  $Af = b$ . A system is overdetermined if the constraints (number of rows  $n$  in  $A$ ) exceed the number of unknowns (number of rows  $r$  in  $f$ ). In general, if in (30)  $r < n$ , the system is overdetermined. In the case of community detection, (1) that defines curvature requires only the first and second derivatives; thus,  $r = 2$ .

$$\begin{bmatrix} d_1 & \dots & \frac{d_1^r}{r!} \\ \dots & \dots & \dots \\ d_n & \dots & \frac{d_n^r}{r!} \end{bmatrix} \begin{bmatrix} F^1(v_c) \\ \dots \\ F^r(v_c) \end{bmatrix} = \begin{bmatrix} F(v_1) - F(v_c) \\ \dots \\ F(v_n) - F(v_c) \end{bmatrix} \tag{30}$$

For this case, the least square approximation can be used. Reduced QR factorization [24] and singular value decomposition (SVD) [35] are well-known methods for approximating the least square solutions; in this work we use QR factorization.

### 4.1.3 Gradient of a node

The vector differential operator  $\nabla$  ('del' or 'nabla') has the following definition:

$$\nabla = \frac{\partial}{\partial x}i + \frac{\partial}{\partial y}j + \frac{\partial}{\partial z}k \tag{31}$$

In the Cartesian space the gradient of  $f(x, y, z)$  is obtained by applying the vector operator  $\nabla$  to the scalar function  $f$  and is defined as:

$$\nabla f(x, y, z) = \frac{\partial f}{\partial x}i + \frac{\partial f}{\partial y}j + \frac{\partial f}{\partial z}k \tag{32}$$

The concept of gradient as used in the context of active contours (3) can be extended to the graph space. In a graph, the dimension of the gradient for a given node is the number of its neighbours. Thus, the gradient of  $F(v)$  in the Figure 6 can be obtained using (33).

$$\nabla F(v) = \frac{\partial F}{\partial v_1}i_1 + \dots + \frac{\partial F}{\partial v_j}i_j + \dots + \frac{\partial F}{\partial v_n}i_n \tag{33}$$

Where  $i_j$  represents the unit vector which is directed to the node  $v_j$ .  $\frac{\partial F}{\partial v_j}$  is calculated from the numerical approximation of the derivative as shown in (34).

$$\frac{\partial F}{\partial v_j} = \frac{F(v) - F(v_j)}{d_j} \tag{34}$$

Therefore, the magnitude of the gradient  $|\nabla F(v)|$  is defined by (35).

$$|\nabla F(v)| = \sqrt{\left(\frac{\partial F}{\partial v_1}\right)^2 + \dots + \left(\frac{\partial F}{\partial v_n}\right)^2} \tag{35}$$

### 4.1.4 Analysing the error for derivative approximations

To analyse the error of derivatives, we generalise two cases of central and non-central nodes into one equation:

$$\begin{bmatrix} L_1 & \dots & \frac{L_1^n}{n!} \\ \dots & & \\ L_n & \dots & \frac{L_n^n}{n!} \end{bmatrix} \begin{bmatrix} F^1(v_c) \\ \dots \\ F^n(v_c) \end{bmatrix} = \begin{bmatrix} F(v_1) - F(v_c) \\ \dots \\ F(v_n) - F(v_c) \end{bmatrix} \tag{36}$$

In the case of a central node  $L_i = d_i$ , and  $L_i = (m + d_i)$  for non-central cases. Every equation in (36), is associated with a Taylor truncation error. This error is committed by replacing  $F(V_i)$  by the first n terms of Taylor expansion, and is modeled by the remainder  $\delta F_i$ :

$$F(v_i) - F(v_c) = L_i F_V^1(v_c) + \dots + \frac{L_i^n}{n!} F_V^n(v_c) + \delta F_i \tag{37}$$

where,  $\delta F_i = \frac{L_i^{(n+1)}}{(n+1)!} F_V^{(n+1)}(C + t_i)$ , and  $t_i$  is in the interval  $(0, L_i)$ . It shows the truncation error and propagates through the system once the derivatives are computed. To observe how it affects the derivatives, these errors are stacked into an error vector  $\delta F = [\delta F_1, \delta F_2, \dots, \delta F_n]^T$ , and (36) is simplified to  $L(f + \delta f) = F + \delta F$ , where  $f$  is the vector of derivatives,  $\delta F$  is the error vector, and  $\delta f$  is the vector of derivative errors. We can then write:

$$\delta f = \delta F \tag{38}$$

$$\delta f = L^{(-1)} \delta F \tag{39}$$

We should mention here that the system proposed in (36) is called well-conditioned when the answer does not change much for small perturbations in the input. The system is ill-conditioned if the matrix L is close to a singular matrix; at least one of the singular values is very close to zero [24]. In this case,  $L^{(-1)}$  magnifies the error and it is necessary to check the error of the derivatives;  $\delta F_i$  can be simplified as:

$$|\delta F_i| \leq e_i = k_i L_i^{n+1}, \text{ where } k_i = \sup_{y \in [C, C+L_i]} \left| \frac{F_v^{(n+1)}}{(n+1)!} \right| \tag{40}$$

Then the error of each derivative follows the equation:

$$|\delta f_i| \leq L^{(-1)} e_i \tag{41}$$

### 4.1.5 Derivative of basic operations

Suppose  $F$  and  $G$  are two functions defined with  $F'(v_c)$  and  $G'(v_c)$  as the first order derivative in node  $v_c$ . Then:

$$\text{Addition rule: } (F \pm G)'(v_c) = F'(v_c) \pm G'(v_c) \tag{42}$$

$$\text{Multiplication rule: } [F(v_c).G(v_c)]' = F'(v_c).G(v_c) + F(v_c).G'(v_c) \tag{43}$$

$$\text{Division rule: } \left[ \frac{F}{G}(v_c) \right]' = \frac{F'(v_c).G(v_c) - F(v_c).G'(v_c)}{[G(v_c)]^2} \tag{44}$$

Proofs are provided in the Appendices A and B.

## 5 Community Detection using differential geometry properties

A community can be seen as a shape in a graph whose nodes are highly connected while their connections to nodes outside the shape are sparse. Since the velocity and its components gradient and curvature are based on derivatives which use the connections between a node on the boundary and its neighbours inside the shape as a basis for deciding the inclusion of a candidate node outside the shape, the approach can be expected to detect good boundaries of shapes.

Before describing the algorithm we need to define a proper  $F$  function.  $F(v_i, v_j)$  represents the distance between  $v_i$  and  $v_j$ . Any suitable distance measure can be chosen for it. The criterion used for  $F(v_i, v_j)$  in this research is the structural equivalence. Nodes are structurally equivalent if they are in the same area of the graph and have the same neighbours. So  $F(v_i, v_j)$  is defined as:

$$F(v_i, v_j) = 1 - \frac{|N(v_i) \cap N(v_j)|}{|N(v_i) \cup N(v_j)|} \quad (45)$$

where  $N(v_i)$  is the set of neighbours of node  $v_i$ , and  $\frac{|N(v_i) \cap N(v_j)|}{|N(v_i) \cup N(v_j)|}$ , or the structural similarity, shows the proportion of the common neighbours.

The algorithm starts from a single node which is assumed to be part of the shape. Initially, the seed node  $v_i$  is considered the current boundary of the shape. A second node  $v_j$ , which has the minimal distance  $F(v_i, v_j)$ , is chosen for inclusion in the shape. As the calculation of the second derivative requires the presence of at least three nodes, a hypothetical node, with the maximum distance of one from the other two nodes, is added, assumed to be part of the shape. This procedure is represented by the line InitialiseCommunity in Algorithm 1.

The shape  $\chi$  initially comprises these three nodes, from which it expands through the inclusion of nodes adjacent to the boundary. Nodes adjacent to the boundary on the outside of the shape are candidates considered for inclusion. Each node  $v_i$  on the boundary which is connected to the candidate node  $v_p$  considers its inclusion based on the velocity function (46).

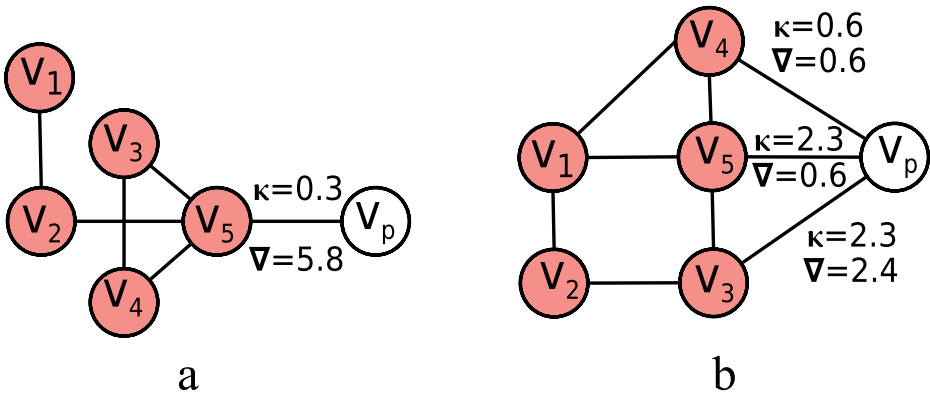
$$s(v_i, v_p) = \frac{\kappa(v_i, v_p)}{1 + \alpha |\nabla F(v_i, v_p)|^2} - \arctan(|F'(v_i, v_p)|) \quad (46)$$

In (46), the curvature is represented by  $\kappa(v_i, v_p)$ , which is defined in (47). The magnitude of the gradient  $|\nabla F(v_i, v_p)|$  describes the difference between  $v_i$  and the candidate node  $v_p$  as calculated by (35). The parameter  $\alpha$  moderates the difference between nodes. The larger alpha, the stricter the criterion for the inclusion of a node. The term  $\arctan(|F'(v_i, v_p)|)$  has been added to map the value of  $|F'(v_i, v_p)|$  to a value between zero and one with the purpose of achieving a negative impact to sudden changes in the derivative of the distance function in order to reduce noise.

$$\kappa(v_i, v_p) = \frac{F''(v_i, v_p)}{\left(1 + (F'(v_i, v_p))^2\right)^{\frac{3}{2}}} \quad (47)$$

As shown in (47), curvature uses the first and second derivatives of the distance function from node  $v_i$  on the boundary to the candidate  $v_p$ . While the gradient bases the decision of

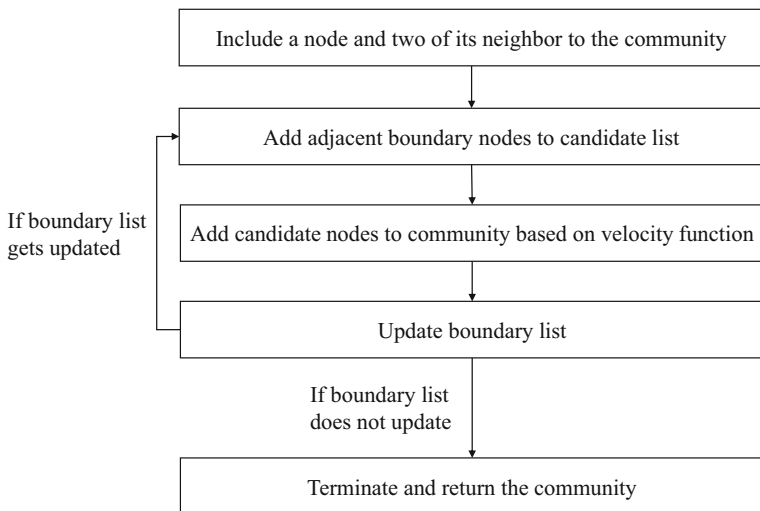




**Figure 7** In both **a** and **b**, shape  $\chi$  consists of  $v_1, v_2, v_3, v_4, v_5$  and  $v_p$  is the candidate for the inclusion. In **a**, values of curvature (47) and gradient (35) from  $v_5$  towards  $v_p$  are shown. In **b**, values of curvature and gradient from  $v_3, v_4$ , and  $v_5$  towards  $v_p$  are shown

an inclusion of node  $v_p$  on the difference between  $v_p$  and the boundary node  $v_i$ , curvature represents the curve of the boundary at  $v_p$  - essentially, ‘concave’ boundaries are more likely to include a node  $v_p$ , because, loosely speaking, it could be seen as ‘enclosed’ by that boundary. In Figure 7, values of curvature and gradient for two simple graphs are shown. By putting the values of curvature and gradient in (46), the velocity from  $v_5$  towards  $v_p$  is a negative value ( $-0.06$ ) in Figure 7a; thus,  $v_p$  will not be included in the community. In Figure 7b, the velocity value towards  $v_p$  is positive for all  $v_3, v_4$ , and  $v_5$ ; therefore, the first one which, according to Algorithm 1, has the chance to include  $v_p$ , will include it and curvature and gradient for the rest of them will not be computed.

Starting from a given seed node, the boundary of a shape moves until the velocity function  $s$  no longer warrants the inclusion of further nodes. Candidate nodes are evaluated from



**Figure 8** Flowchart of DCD

all nodes on the boundary they are connected to, but the evaluation stops as soon as one of the boundary nodes favours the inclusion of the node. This means that most of the time, the algorithm achieves a significantly better run time than required by its worst case complexity. Figure 9 illustrates an example of the proposed algorithm. The diagram in Figure 8 shows the process of DCD.

In (46), the velocity function has only one parameter,  $\alpha$ . To give the user control over size and quality of the desired communities  $\alpha$  is added to the inclusion criteria. The larger  $\alpha$  is, the stronger the effect of the gradient, and therefore the sharper the edge.

---

**Algorithm 1** Derivative based community detection.

---

```

Input:  $seed, \alpha$ 
Queue  $Boundary = seed$ 
Set  $C \leftarrow \text{InitialiseCommunity}(seed)$ 
Boolean  $switch = false$ 
while  $switch \neq true$  do
   $switch = true$ 
  Node  $v_x \leftarrow \text{Deque}(Boundary)$ 
   $candidate\ list \leftarrow \text{OutsideNeighbours}(v_x)$ 
  for every  $v_p$  in  $candidate\ list$  do
    if  $\text{Velocity}(v_x, v_p, \alpha) > 0$  then
       $C \leftarrow C \cup \{v_p\}$ 
       $Boundary \leftarrow \text{UpdateBoundary}()$ 
       $switch = false$ 
    end if
  end for
end while
return  $C$ 

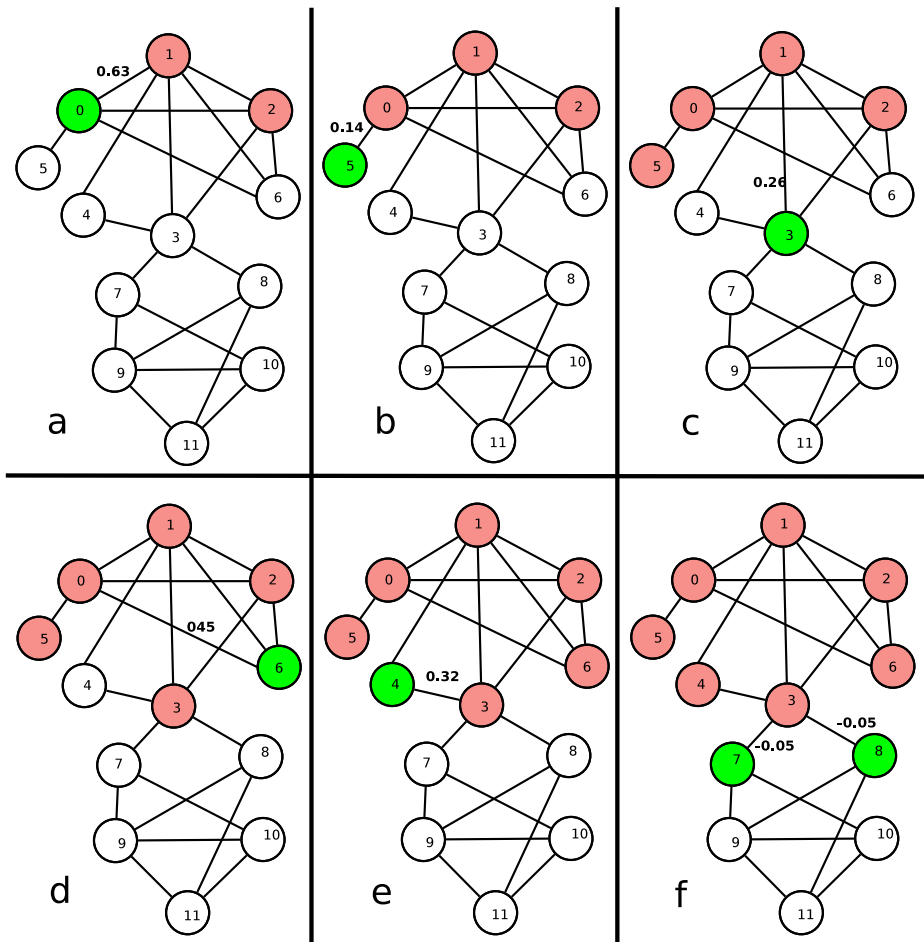
```

---

## 6 Tracking communities

To our knowledge, the proposed model is the only local community detection method which is able to track temporal smoothness changes in a network. In temporal smoothness, there is a stream of atomic changes. The community updates itself triggered by a change. Our model, currently, considers addition and removal of edges and addition of nodes. Dealing with node removal, due to its complex nature, is left for future work. To affect a community, a change must happen either inside or in the neighbourhood of the community. The steps taken to deal with different types of changes are:

- **Edge addition:** When an edge is added there are three possibilities:
  - The new edge is inside the community. This strengthens the existing community. Therefore, no adjustment is necessary.
  - The new edge is outside the community. Following the same logic, no change is necessary.
  - The new edge connects a node inside the community to a node outside the community. The velocity to the node which is located outside the community has to be calculated from all of its neighbours inside the community. If the velocity is positive, the node is included.



**Figure 9** The red nodes show the current community and the green nodes are candidates for inclusion. The number on the edges shows the velocity from a node to the candidate. When the velocity towards all neighbouring nodes is negative the algorithm stops

#### – Edge deletion:

- An edge inside the community is removed. The nodes on both ends of the removed edge are checked by calculating the velocities from their neighbouring nodes.
- An edge outside the community is removed. This does not affect the community and no adjustment is necessary.
- An edge that connects a node inside the community to a node outside is removed. Removing the edge only makes the community stronger, therefore no adjustment is necessary.

#### – Node addition: We have the following possibilities when a node is added to the graph:

- One or more edges connect the node to the community. The velocity from each of the neighbouring nodes to the new candidate are calculated. In case the velocity is positive, the node is included and the algorithm stops.
- The additional node is not connected to the community. The community is not affected and no adjustment is necessary.

## 7 Time Complexity

This research has two sections: finding a local community around a seed node and tracking a local community given a stream of atomic changes. The time complexity analysis of both section are analyses.

### 7.1 Time complexity of finding a local community

The main computation in DCD is to find the first and second derivatives for the nodes on the boundary. These derivatives only consider the inside neighbours which in average their number is less than the average of node's degree. Because average degree considers outside neighbours, too. For example in Figure 9f, node 3 has three neighbours inside the community and two neighbours outside of the community.

If the average degree of the nodes on the boundary is  $d$ , and there are  $m$  nodes on the boundary then the calculation of velocity require  $O(md^2)$  calculations where  $d^2$  is the time complexity for QR factorization to approximate the first and the second derivatives. Knowing the fact that every node inside the community was once on the boundary, if the size of community is  $n$ , where  $m \leq n$ , then time complexity of DCD should be  $O(nd^2)$ . However, the real time complexity is more than this because of the re-calculation of the velocity function for some nodes. This happens when the current node  $v_c$  that is located on the boundary includes an outside neighbour into the community (because velocity from  $v_c$  to the outside neighbour was positive). This means all of the inside neighbours of  $v_c$  which are on the boundary has to update their velocity values. The number of neighbours which are on the boundary is less than  $d$ . Therefore, time complexity of DCD will be  $O(nd^3)$  where  $n$  is the size of community and  $d$  is the average degree of the nodes inside the community.

### 7.2 Time complexity of tracking a local community

To track the community after a single atomic change, according to Section 6, the velocity of a constant number of nodes needs to be calculated. Therefore, the time complexity of updating the community after an atomic change is  $O(d^2)$  where  $d$  is the average degree of the nodes, and  $d^2$  the time to approximate derivatives. The efficient time complexity of DCD makes it ideal for tracking local communities in large scale dynamic networks.

## 8 Experimental evaluation

### 8.1 Finding local communities

This work proposes a framework for approximating derivatives in a graph space to be used for community detection. The goal of this evaluation is to demonstrate that the derivatives can be used to identify good communities as defined by a commonly used quality measure

for communities in graphs. The aim is also to show that these communities are of comparable, sometimes better, quality as communities produced by two well-known community detection methods. We used Local Spectral Partitioning (LSP) [39], parameterised as recommended by Jeub et al. [27], and Local Geodesic Spreading (LGS) [3] as benchmarks for comparison. The graphs used in the experimentation are Facebook graph FB-JHK of John Hopkins University with 5180 nodes and 186572 edges, and FB-CALTC of California Institute of Technology with 769 nodes and 33312 edges, both captured in September 2005 and are part of the FACEBOOK100 dataset [59].

**Quality of communities** The three most widely known measures for determining the quality of local communities are relative density, intra-cluster density and conductance. Relative density is the ratio between the number of intra-cluster edges and the number of edges that connect the community to the rest of the graph. Intra-cluster density [38, 44] is the fraction of the number of edges inside the community as defined as:

$$\text{Internal density}(C) = \frac{e_C}{\frac{n_C(n_C-1)}{2}} \quad (48)$$

where  $e_C$  is the number of edges in community  $C$  and  $n_C$  is the number of nodes in  $C$ .  $\frac{n_C(n_C-1)}{2}$  shows maximum possible number of edges within community  $C$ . Higher internal density means better quality for the community.

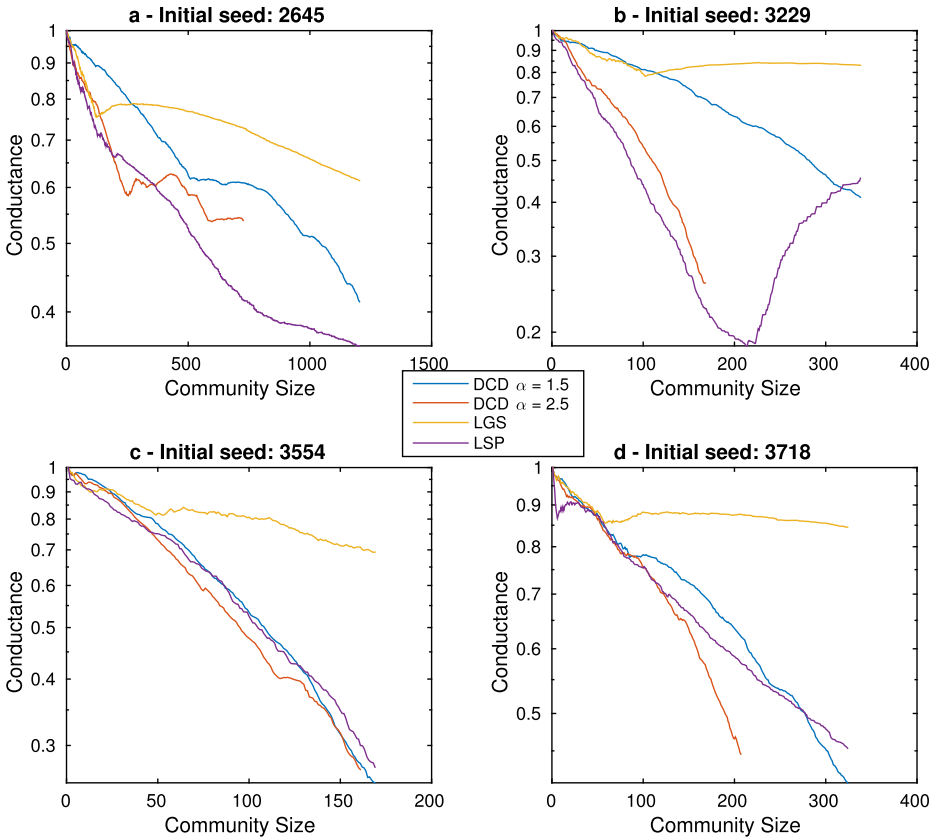
Conductance is defined as:

$$\text{Conductance}(C) = \frac{\text{vol}(C, \bar{C})}{\min(\text{vol}(C, G), \text{vol}(\bar{C}, G))} \quad (49)$$

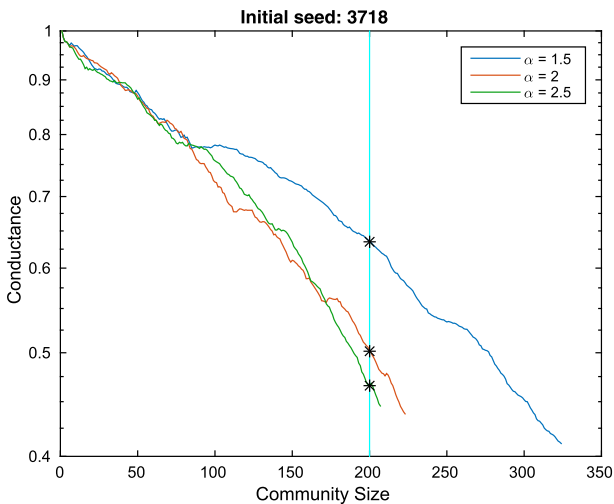
In (49),  $C$  is the set of nodes which comprise the community,  $\bar{C} = V - C$  denotes the nodes in the graph which are not in  $C$ , and  $\text{vol}(C_1, C_2) = \sum_{i \in C_1} \sum_{j \in C_2} A_{ij}$ .  $A$  is the adjacency matrix.  $\text{Conductance}(C)$  has a lower value when the community is loosely connected to the rest of the graph. Therefore, the lower the conductance, the higher the quality of the community. Following the practice of a number of recent studies of significance [27, 37, 39], we use conductance as the main quality measure in this paper. We also compare the results of DCD with baselines using in

## 8.2 Tracking local communities

To evaluate DCD for tracking communities, the dynamic communities network generator by Gorke and et al. [25] is used. The benefit of their clustered network generator is its capability to generate ground truth dynamic communities, given an atomic change stream. The stream of atomic changes is generated in a way that the community label of every newly added node is known. The ground truth data can be compared against our method's result. Since our method is the first local community detection method of its kind which can track local communities given a stream of changes, we only compared it against ground truth data. In this experiment, several networks with 1000 nodes and 5 communities with different intra-cluster and inter-cluster edge probabilities are generated. Then, 200 nodes are added to the network through a stream of atomic changes. DCD tracks and maintains each of the communities. Since it is known a priori which cluster every newly added node belongs to, we report precision, recall, and F1 score for different scenarios in Section 9. Although convergence of the algorithm cannot be mathematically proven, in all experiments we have observed that as long as the initial node is selected inside the community, the algorithm terminates and separate the community.



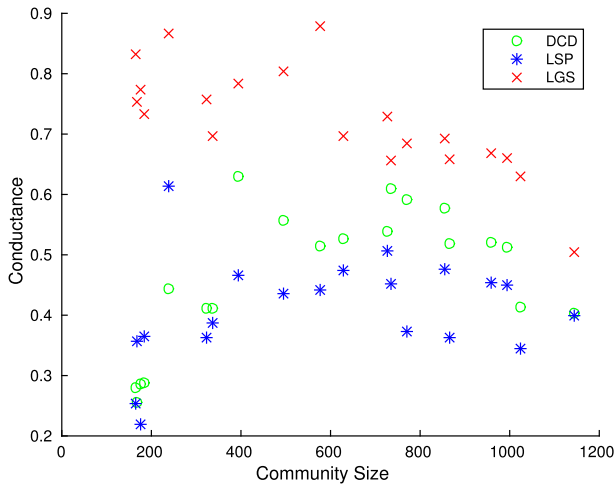
**Figure 10** Conductance plot for different methods and different starting nodes in FB-JHK



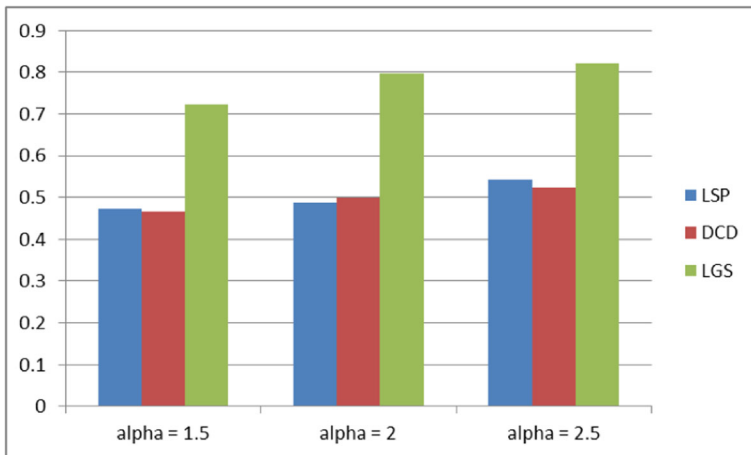
**Figure 11** Effect of  $\alpha$  on finding local communities around a seed node in FB-JHK

**Table 1** Effect of  $\alpha$  on the size of detected communities in FB-JHK for 20 different initial random seeds

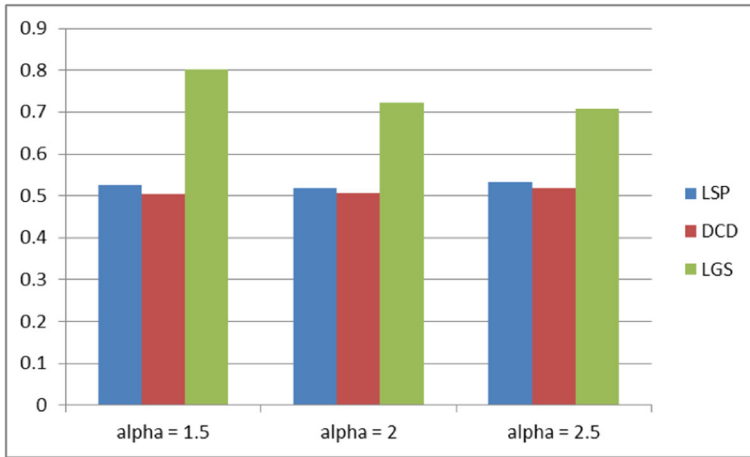
$\alpha$	1.5	2	2.5
Average size	588.2	242.8	173.1



**Figure 12** Conductance of the different detected communities in FB-JHK where  $\alpha = 1.5$



**Figure 13** Average conductances of the communities in FB-JHK

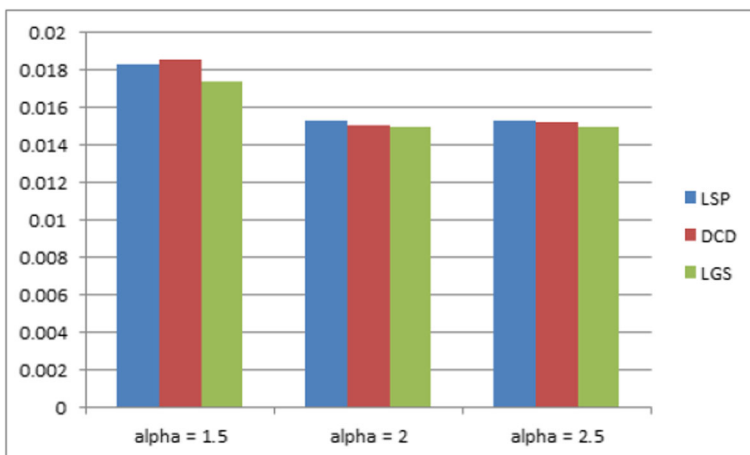


**Figure 14** Average conductances of the communities in FB-CALTC

## 9 Results

### 9.1 Finding local communities

Comparing the outcome of LSP and derivative-based community detection (DCD) has its challenges because both methods depend on a parameter which leads to different combinations of quality and size in the communities detected. The teleportation parameter of LSP defines the type of community being developed. In the experiments, we ran LSP with teleportation set to 20 equally spaced values as explained by Jeub [27]. The parameter  $\alpha$  in



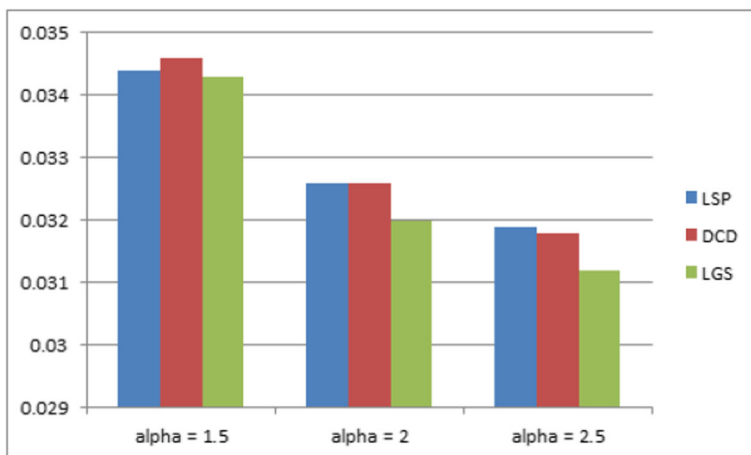
**Figure 15** Average internal density of the communities in FB-JHK



DCD defines the stringency of the inclusion criterion, with larger values being more restrictive. Unlike LSP, DCD stops when according to (46), no further candidate nodes qualify for inclusion.

In Figure 10, we included the progress of the one among the 20 LSP instances that produced the community with the best conductance (regardless of the size of the community) for the JHK-FB network. LGS, which is based on PageRank, has no parameters except the seed node, hence there is no choice in the result to include. Because of the variation in the parameter  $\alpha$ , we included two result graphs for DCD. Figure 10a-d represent trials starting from four different seed nodes and were chosen because they are representative of the different behaviours of the algorithms. Figure 10a shows a case where LSP outperforms all others except DCD with  $\alpha = 2.5$  when the community has a size of around 200 nodes. In Figure 10b the smaller communities found by LSP are of slightly better quality than those of DCD, but DCD with  $\alpha = 1.5$  discovers a community with around 330 nodes with better conductance. In Figure 10c, the performances of LSP and DCD are almost equivalent – in most cases, DCD with  $\alpha = 2.5$  produces slightly better quality than LSP, but all three algorithms produce similar results. In Figure 10d, DCD with  $\alpha = 2.5$  produces considerably better quality for smaller communities while DCD with  $\alpha = 1.5$  shows better conductance for larger communities. LGS is not a competitive algorithm in any of the cases examined.

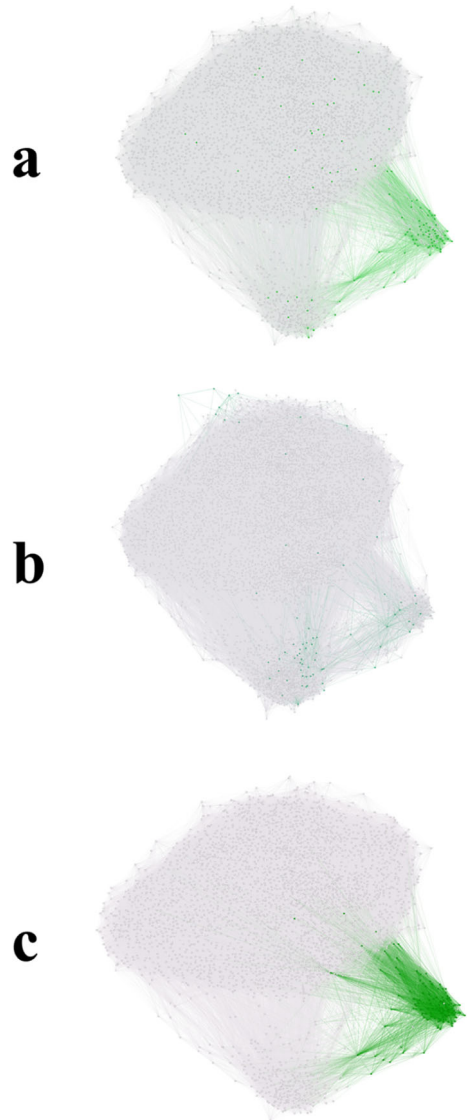
The results shown in Figure 10 illustrate the difference in performance of DCD that the parameter  $\alpha$  entails. This raises the question how to identify the best setting for  $\alpha$ . Further investigations, illustrated in Figure 11, show that smaller values of  $\alpha$  lead to the detection of larger communities, while larger values of  $\alpha$  discover small-size communities. Because larger values of  $\alpha$  restrict the inclusion of new nodes earlier, the algorithm stops at a smaller community size. Table 1 shows the average sizes of the communities detected with different values of  $\alpha$ . The quality of the community found, large or small, depends on the initial seed. This property is common to DCD and most other methods, including LSP and LGS (Fig. 12).



**Figure 16** Average internal density of the communities in FB-CALTC

Figures 13 and 14 compare the average conductances achieved by the different algorithms for a community of a particular size starting from 20 different random seeds. The size is dictated by the number of nodes included by DCD with the value of  $\alpha$  given. Since several restarts were used, the size is not exactly identical in each of the restarts, but for each restart, the community with an equivalent size produced by LSP and LGS were chosen to calculate the average conductivity. For LSP, the trials were repeated with each of the 25 parameters for teleportation and the average is reported. Figures 13 and 14 show the conductance of the detected communities for DCD, LSP, and LGS for the 20 random seeds in

**Figure 17** Finding the local community of node 3554 in FB-JHK using **a**:LGS, **b**:LSP, and **c**:DCD



**Table 2** Different parameterisation for intra-cluster ( $P_{in}$ ) and inter-cluster ( $P_{out}$ ) probabilities

	Scenarios						
	A	B	C	D	E	F	G
$P_{in}$	0.8	0.8	0.8	0.7	0.7	0.6	0.5
$P_{out}$	0.1	0.3	0.2	0.2	0.3	0.3	0.3

FB-JHK and FB-CALTC. As it can be seen, DCD has the best performance followed closely by LSP and then with some margin is the LGS. Figures 15 and 16 illustrate the average internal density of the detected communities. For lower values of  $\alpha$  DCD outperforms both LSP and LGS. However, for the greater values of  $\alpha$ , which according to Figure 11 results into smaller community sizes, LSP outperforms DCD and LGS remains the weakest among the three methods.

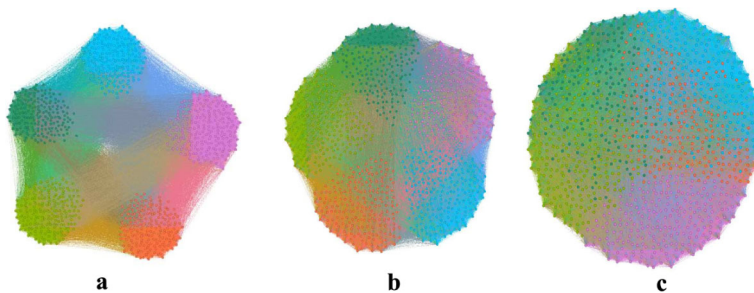
Figure 17 illustrates the detected communities around node 3554 using LGS, LSP, and DCD respectively. For the sake of visualisation, nodes with the degree less than 15 were ignored. The experiments show LSP, unlike DCD and LGS, tends to include a considerable number of nodes with the relatively low degree.

## 9.2 Tracking local communities

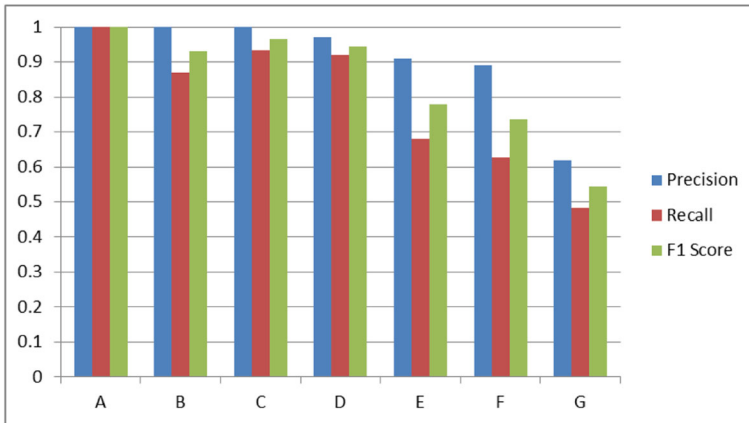
To test the performance of DCD for tracking local communities, seven different scenarios with ground truth dynamic communities were generated. Each network initially has 1000 nodes with an average degree of 30. Then, 200 nodes are successively added to the network. The seven experiments differ in their probabilities of inter-cluster and intra-cluster edges. Experiments are labeled in alphabetical order. Their parameterisations are shown in Table 2.

Figure 18a to c show the structure of the networks in scenarios A, D, and G from Table 2. In Figure 18a, the five ground-truth communities are very well defined without much overlaps. However, since the  $P_{intra-cluster}$  increases as we go through subsequent scenarios, the quality of the communities drops. In Figure 18c, the communities due to the high value of  $P_{intra-cluster}$ , the communities are intertwined.

The precision, recall, and F1 scores for each of the experiments are shown in Figure 19. As the probability of edges within clusters decreases and the probability of edges between clusters increases, tracking communities becomes more difficult and the accuracy decreases. For well-defined communities, DCD performs well.



**Figure 18** The structure of the networks for different scenarios in Table 2. **a:** Scenario A, **b:** Scenario D, **c:** Scenario G



**Figure 19** Precision, recall, and F1 score for each scenario

## 10 Conclusion and further work

This study has introduced a new local community detection algorithm based on derivatives. It has necessitated the definition of an approach for approximating derivatives in graphs. The method has been shown to perform on a par with or better than a well-known local community detection method, and both methods have similar computational complexity. DCD offers the advantage of stopping when all nodes that qualify have been included in the community. The  $\alpha$  parameter offers a means of controlling the size of the desired community. Although DCD does not consistently outperform LPS, it has significant potential as a local community detection method. Given its inspiration from Active Contours, many of the improvements that have been applied to Active Contours over time can also be adapted to DCD, for example Chan and Vese's work [8] which serves to reduce the sensitivity to noise, can be added in the form of considering the density of the edges inside the community. One direction of future research is to investigate the problem of tracking local communities using different dynamic network generators with ground truth communities. One significant advantage of DCD is its success in tracking dynamic communities in a computationally cheap fashion, because the algorithm only needs to recalculate the 'velocity' function (46) for the nodes that have changed if they are located inside the community or in the vicinity of the boundary. This is a considerable contribution given the lack of local community tracking methods and the prevalence of very large graphs whose communities matter in a social and commercial context. The application of derivatives to graph space in itself has a multitude of applications, such as using features of differential geometry for visualisation or using heat diffusion in thermodynamics for modeling information diffusion in social networks.

## Appendix A: Error analysis

Suppose  $F$  and  $G$  are two functions defined with  $F'(v_c)$  and  $G'(v_c)$  as the first order derivative in node  $C$ . Then:

$$\text{Addition rule: } (F \pm G)'(v_c) = F'(v_c) \pm G'(v_c) \quad (50)$$

$$\text{Multiplication rule: } [F(v_c).G(v_c)]' = F'(v_c).G(v_c) + F(v_c).G'(v_c) \tag{51}$$

$$\text{Division rule: } \left[\frac{F}{G}(v_c)\right]' = \frac{F'(v_c).G(v_c) - F(v_c).G'(v_c)}{[G(v).G(v_c)]^2} \tag{52}$$

Proof: Suppose  $D$  is the derivative matrix in (30). We define  $D_1^{(-1)}$  as the first row of matrix  $D^{(-1)}$ , and  $H = F + G$ :

$$DH^{(n)} = H(v) - H(v_c); \text{ where } H^{(n)} \text{ is the vector of derivatives}$$

$$H'(v_c) = D_1^{(-1)}[H(v) - H(v_c)] \tag{53}$$

$$H'(v_c) = D_1^{(-1)}[F(v) + G(v) - F(v_c) - G(v_c)] \tag{54}$$

$$H'(v_c) = D_1^{(-1)}[F(v) - F(v_c)] + D_1^{(-1)}[G(v) - G(v_c)] = F'(v_c) + G'(v_c) \tag{55}$$

Differentiation rule’s proof is similar to addition.

Multiplication rule: We define  $H = F.G$  where  $.$  is the inner product operator.

$$H'(v_c) = D_1^{(-1)}[H(v) - H(v_c)] \tag{56}$$

$$H'(v_c) = D_1^{(-1)}[F(v).G(v) - F(v_c).G(v_c)] \tag{57}$$

$$H'(v_c) = D_1^{(-1)}[F(v).G(v) - F(v_c).G(v) + F(v_c).G(v) - F(v_c).G(v_c)] \tag{58}$$

$$D_1^{(-1)}[G(v).(F(v) - F(v_c))] + D_1^{(-1)}[F(v_c).(G(v) - G(v_c))] \tag{59}$$

$$H'(v_c) = F'(v_c).G(v) + F(v_c).G'(v_c) \tag{60}$$

Division rule:

$$H'(v_c) = D_1^{(-1)}\left[\frac{F(v)}{G(v)} - \frac{F(v_c)}{G(v_c)}\right] \tag{61}$$

$$H'(v_c) = D_1^{(-1)}\left[\frac{F(v).G(v_c) - F(v_c).G(v_c) + F(v_c).G(v_c) - F(v_c).G(v)}{G(v).G(v_c)}\right] \tag{62}$$

$$H'(v_c) = \frac{F'(v_c).G(v_c)}{G(v).G(v_c)} - \frac{F(v_c).G'(v_c)}{G(v).G(v_c)} \tag{63}$$

$$H'(v_c) = \frac{F'(v_c).G(v_c) - F(v_c).G'(v_c)}{G(v).G(v_c)} \tag{64}$$

Depending on the neighbors of node  $C$ , the term  $G(v)$ , appeared in the right hand side of multiplication and division rules, can be approximated differently. For a simple case where node  $C$  is connected to just one node  $V$ , multiplication is rewritten:

$$H'(v_c) = F'(v_c).[G(v_c) + LG'(v_c)] + F(v_c).G'(v_c) \tag{65}$$

$$H'(v_c) = F'(v_c).G(v_c) + F(v_c).G'(v_c) + LF'(v_c).G'(v_c) \tag{66}$$

and division rule is simplified:

$$H'(v_c) = \frac{F'(v_c).G(v_c) - F(v_c).G'(v_c)}{[G(v_c) + LG'(v_c)].G(v_c)} \quad (67)$$

$$H'(v_c) = \frac{F'(v_c).G(v_c) - F(v_c).G'(v_c)}{G^2(v_c) + L.G(v_c).G'(v_c)} \quad (68)$$

## Appendix B: List of symbols

$C$	community
$t$	time step
$\nabla$	differential operator
$\kappa$	curvature
$\partial$	partial derivative
$\mathbf{n}$	normal direction
$V$	set of graph nodes
$G$	graph
$\chi$	graph shape
$E$	set of graph nodes
$\nu$	set of shape nodes
$\xi$	set of shape edges
$O$	truncation error
$F$	similarity function
$d$	node difference
$e$	approximation error
$N$	set of a node's neighbours
$s$	velocity

## References

1. Ahn, Y.-Y., Bagrow, J.P., Lehmann, S.: Link communities reveal multiscale complexity in networks. *Nature* **466**(7307), 761–764 (2010)
2. Andersen, R., Lang, K.J.: Communities from seed sets. In: Proceedings of the 15th international conference on World Wide Web, pp. 223–232. ACM (2006)
3. Bagrow, J.P., Bollt, E.M.: Local method for detecting communities. *Phys. Rev. E* **72**(4), 046108 (2005)
4. Bansal, N., Blum, A., Chawla, S.: Correlation clustering. *Mach. Learn.* **56**(1-3), 89–113 (2004)
5. Brandes, U., Delling, D., Gaertler, M., Görke, R., Hofer, M., Nikoloski, Z., Wagner, D.: On modularity- $\eta$ -completeness and beyond. *Citeseer* (2006)
6. Canu, M., Lesot, M.-J., d'Allonnes, A.R.: Vertex-centred method to detect communities in evolving networks. In: International workshop on complex networks and their applications, pp. 275–286. Springer (2016)
7. Caselles, V., Kimmel, R., Sapiro, G.: Geodesic active contours. *Int. J. Comput. Vis.* **22**(1), 61–79 (1997)
8. Chan, T.F., Vese, L., et al.: Active contours without edges. *IEEE Trans. Image. Process.* **10**(2), 266–277 (2001)
9. Charikar, M., Guruswami, V., Wirth, A.: Clustering with qualitative information. In: 44th Annual IEEE symposium on foundations of computer science, 2003. Proceedings, pp. 524–533. IEEE (2003)
10. Chen, J., Yuan, B.: Detecting functional modules in the yeast protein–protein interaction network. *Bioinformatics* **22**(18), 2283–2290 (2006)


11. Chen, J., Zaiane, O.R., Goebel, R.: Detecting communities in large networks by iterative local expansion. In: International conference on computational aspects of social networks, 2009. CASON'09, pp. 105–112. IEEE (2009)
12. Clauset, A.: Finding local community structure in networks. *Phys. Rev. E* **72**(2), 026132 (2005)
13. Clauset, A., Newman, M.E., Moore, C.: Finding community structure in very large networks. *Phys. Rev. E* **70**(6), 066111 (2004)
14. Coleman, J.S. et al.: Introduction to mathematical sociology. London Free Press, Glencoe (1964)
15. Courant, R., Friedrichs, K., Lewy, H.: Über die partiellen differenzgleichungen der mathematischen physik. *Math. Ann.* **100**(1), 32–74 (1928)
16. Danon, L., Diaz-Guilera, A., Duch, J., Arenas, A.: Comparing community structure identification. *J. Stat. Mech.: Theory. Exper.* **2005**(09), P09008 (2005)
17. Diao, P., Guillot, D., Khare, A., Rajaratnam, B.: Differential calculus on graphon space. *J. Comb. Theory. Series. A* **133**, 183–227 (2015)
18. Du, H., Feldman, M.W., Li, S., Jin, X.: An algorithm for detecting community structure of social networks based on prior knowledge and modularity. *Complexity* **12**(3), 53–60 (2007)
19. Falkowski, T.: Community analysis in dynamic social networks. Sierke (2009)
20. Farhangi, M.M., Frigui, H., Bert, R., Amini, A.A.: Incorporating shape prior into active contours with a sparse linear combination of training shapes: application to corpus callosum segmentation. In: 2016 IEEE 38th Annual international conference of the engineering in medicine and biology society (EMBC), pp. 6449–6452. IEEE (2016)
21. Farhangi, M.M., Frigui, H., Seow, A., Amini, A.A.: 3-d active contour segmentation based on sparse linear combination of training shapes (scots). *IEEE Trans. Med. Imaging.* **36**(11), 2239–2249 (2017)
22. Fortunato, S.: Community detection in graphs. *Phys. Rep.* **486**(3), 75–174 (2010)
23. Friedman, J., Tillich, J.-P.: Calculus on graphs. arXiv preprint cs/0408028 (2004)
24. Golub, G.H., Van Loan, C.F.: Matrix computations, vol 3. JHU Press (2012)
25. Görke, R., Kluge, R., Schumm, A., Staudt, C., Wagner, D.: An efficient generator for clustered dynamic random networks. In: Design and analysis of algorithms - first mediterranean conference on algorithms, MedAlg 2012, Kibbutz Ein Gedi, Israel, December 3-5, 2012. Proceedings, volume 7659 of lecture notes in computer science (LNCS), pp. 219–233. Springer International Publishing, Switzerland (2012)
26. Hu, Y., Yang, B., Lv, C.: A local dynamic method for tracking communities and their evolution in dynamic networks. *Knowl-Based. Syst.* **110**, 176–190 (2016)
27. Jeub, L.G., Balachandran, P., Porter, M.A., Mucha, P.J., Mahoney, M.W.: Think locally, act locally: detection of small, medium-sized, and large communities in large networks. *Phys. Rev. E* **91**(1), 012821 (2015)
28. Johnson, S.C.: Hierarchical clustering schemes. *Psychometrika* **32**(3), 241–254 (1967)
29. Khorasgani, R.R., Chen, J., Zaiane, O.R.: Top leaders community detection approach in information networks. In: 4th SNA-KDD workshop on social network mining and analysis (2010)
30. Kleinberg, J.M.: Authoritative sources in a hyperlinked environment. *J. ACM* **46**(5), 604–632 (1999)
31. Kottak CP: Cultural anthropology: appreciating cultural diversity. McGraw-Hill (2011)
32. Krishnamurthy, B., Wang, J.: On network-aware clustering of Web clients. *ACM SIGCOMM. Comput. Commun. Rev.* **30**(4), 97–110 (2000)
33. Lancichinetti, A., Fortunato, S.: Community detection algorithms: a comparative analysis. *Phys. Rev. E* **80**(5), 056117 (2009)
34. Lancichinetti, A., Fortunato, S., Kertész, J.: Detecting the overlapping and hierarchical community structure in complex networks. *J. Phys.* **11**(3), 033015 (2009)
35. Lawson, C.L., Hanson, R.J.: Solving least squares problems, vol 15. SIAM (1995)
36. Leskovec, J., Kleinberg, J., Faloutsos, C.: Graphs over time: densification laws, shrinking diameters and possible explanations. In: Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining, pp. 177–187. ACM (2005)
37. Leskovec, J., Lang, K.J., Dasgupta, A., Mahoney, M.W.: Community structure in large networks: natural cluster sizes and the absence of large well-defined clusters. *Internet. Math.* **6**(1), 29–123 (2009)
38. Leskovec, J., Lang, K.J., Mahoney, M.: Empirical comparison of algorithms for network community detection. In: Proceedings of the 19th international conference on World wide Web, pp. 631–640. ACM (2010)
39. Mahoney, M.W., Orecchia, L., Vishnoi, N.K.: A local spectral method for graphs: with applications to improving graph partitions and exploring data graphs locally. *J. Mach. Learn. Res.* **13**(1), 2339–2365 (2012)
40. McAuley, J., Leskovec, J.: Discovering social circles in ego networks. *ACM Trans. Knowl. Discov. Data. (TKDD)* **8**(1), 4 (2014)

41. Ng, A.Y., Jordan, M.I., Weiss, Y., et al.: On spectral clustering: analysis and an algorithm. *Adv. Neural. Inf. Process. Syst.* **2**, 849–856 (2002)
42. Nguyen, N.P., Dinh, T.N., Tokala, S., Thai, M.T.: Overlapping communities in dynamic networks: their detection and mobile applications. In: *Proceedings of the 17th annual international conference on mobile computing and networking*, pp. 85–96. ACM (2011)
43. Palla, G., Derényi, I., Farkas, I., Vicsek, T.: Uncovering the overlapping community structure of complex networks in nature and society. *Nature* **435**(7043), 814–818 (2005)
44. Radicchi, F., Castellano, C., Ceconi, F., Loreto, V., Parisi, D.: Defining and identifying communities in networks. *Proc. Natl. Acad. Sci. USA* **101**(9), 2658–2663 (2004)
45. Rand, W.M.: Objective criteria for the evaluation of clustering methods. *J. Am. Stat. Assoc.* **66**(336), 846–850 (1971)
46. Rives, A.W., Galitski, T.: Modular organization of cellular networks. *Proc. Natl. Acad. Sci.* **100**(3), 1128–1133 (2003)
47. Rigi, M.A., Moser, I., Rigi, S., Liu, C.: Re-imaginig the networks: detecting local communities in networks by approximating derivatives in graph space. In: *Proceedings of the 2017 IEEE/ACM international conference on advances in social networks analysis and mining 2017, ASONAM '17*, pp. 974–981. ACM, New York (2017)
48. Samie, M.E., Hamzeh, A.: Community detection in dynamic social networks: a local evolutionary approach. *J Inf Sci*, 0165551516657717 (2016)
49. Schaeffer, S.E.: Graph clustering. *Comput. Sci. Rev.* **1**(1), 27–64 (2007)
50. Serrano, M., Boguñá, M., Pastor-Satorras, R., Vespignani, A.: Large scale structure and dynamics of complex networks: from information technology to finance and natural sciences (2007)
51. Shang, J., Liu, L., Li, X., Xie, F., Wu, C.: Targeted revision: a learning-based approach for incremental community detection in dynamic networks. *Physica. A.: Stat. Mech. Appl.* **443**(Supplement C), 70–85 (2016)
52. Solomon, J.: Pde approaches to graph analysis. *arXiv:1505.00185* (2015)
53. Spiliopoulou, M., Ntoutsis, I., Theodoridis, Y., Schult, R.: Monic: modeling and monitoring cluster transitions. In: *Proceedings of the 12th ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 706–711. ACM (2006)
54. Sun, J., Faloutsos, C., Papadimitriou, S., Yu, P.S.: Graphscope: parameter-free mining of large time-evolving graphs. In: *Proceedings of the 13th ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 687–696. ACM (2007)
55. Takaffoli, M., Fagnan, J., Sangi, F., Zaiiane, O.R.: Tracking changes in dynamic information networks. In: *2011 International conference on computational aspects of social networks (CASoN)*, pp. 94–101. IEEE (2011)
56. Takaffoli, M., Rabbany, R., Zaiiane, O.R.: Incremental local community identification in dynamic social networks. In: *Proceedings of the 2013 IEEE/ACM international conference on advances in social networks analysis and mining*, pp. 90–94. ACM (2013)
57. Tong, H., Papadimitriou, S., Sun, J., Yu, P.S., Faloutsos, C.: Colibri: fast mining of large static and dynamic graphs. In: *Proceedings of the 14th ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 686–694. ACM (2008)
58. Toyoda, M., Kitsuregawa, M.: Creating a Web community chart for navigating related communities. In: *Proceedings of the 12th ACM conference on hypertext and hypermedia, HYPERTEXT '01*, pp. 103–112. ACM, New York (2001)
59. Traud, A.L., Mucha, P.J., Porter, M.A.: Social structure of facebook networks. *Physica. A.: Stat. Mech. Appl.* **391**(16), 4165–4180 (2012)
60. Xu, T., Zhang, Z., Yu, P.S., Long, B.: Generative models for evolutionary clustering. *ACM Trans. Knowl. Discov. Data. (TKDD)* **6**(2), 7 (2012)
61. Yang, T., Chi, Y., Zhu, S., Gong, Y., Jin, R.: Detecting communities and their evolutions in dynamic social networks—a bayesian approach. *Mach. Learn.* **82**(2), 157–189 (2011)
62. Yoshida, T.: Toward finding hidden communities based on user profile. *J. Intell. Inf. Syst.* **40**(2), 189–209 (2013)

**Publisher's note** Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.



## Affiliations

M. Amin Rigi<sup>1</sup>  · Irene Moser<sup>1</sup> · M. Mehdi Farhangi<sup>2</sup> · Chengfei Lui<sup>1</sup>

Irene Moser  
imoser@swin.edu.au

M. Mehdi Farhangi  
m0farh03@louisville.edu

Chengfei Lui  
clui@swin.edu.au

<sup>1</sup> Swinburne University of Technology, Melbourne, Australia

<sup>2</sup> University of Louisville, Louisville, KY, USA