Derivatives in Graph Space with Applications for Finding and Tracking Local Communities

M. Amin Rigi, Irene Moser, and M. Mehdi Farhangi

Abstract Community detection in networks has gained a lot of attention especially after emergence of online social networks. Community detection methods in networks can be classified into two domains: global methods and local methods. Global methods need the whole information of the network, whereas the local ones need information of a certain area of the network where they want to discover communities. Real-world social networks are typically very large, making the global community detection methods impractical due to the computation expenses. Therefore, local community detection algorithms, which are requiring less computation and space, have met with renewed interest. In this research two derivative-based methods for finding and tracking local communities are proposed. Mapping the concepts of derivatives into graph space in a practical manner poses few challenges. For instance, in Euclidean space, every point has three dimensions, whereas in graph space the dimension (or degree) of every node can be different. Firstly, we propose a general framework for finding derivatives in graph space. This mentioned framework enables us to bring derivative-based methods into graph theory. Secondly, inspired by the active contour algorithm in computer vision domain, we propose a local derivative-based community detection method. The proposed method is built upon concepts of curvature and gradient of the community's boundary. Curvature and gradient comprise a velocity function to determine whether the boundary should expand to include a candidate node in its vicinity. Finally, based on derivative-based concept of surface tension in chemistry, we propose a model for tracking local communities in dynamic networks where new nodes/edges are added in a stream of atomic changes. The binding forces between the molecules of the same liquid substance give them shape with the minimum surface tension. That is to say, if

M. A. Rigi (✉) · I. Moser
Swinburne University of Technology, Melbourne, VIC, Australia
e-mail: mrigi@swin.edu.au; imoser@swin.edu.au

M. M. Farhangi
University of Louisville, Louisville, KY, USA
e-mail: m0farh03@louisville.edu

P. Karampelas et al. (eds.), *From Security to Community Detection in Social Networking Platforms*, Lecture Notes in Social Networks,
https://doi.org/10.1007/978-3-030-11286-8_4

molecules of the same substance are added to the community, the surface tension should not increase. In the network context, if a node can be added to a community it reduces the surface tension of the community. Experimental results validate the superiority of the proposed methods.

Keywords Local community detection · Tracking community · Derivatives in networks · Surface tension

1 Introduction

One phenomenon in nature that scientist through the history tried to explain and predict is the community. Analysing communities is a principal topic in sociology. There exist many systems in the world that can be represented with networks where connections, or links, show relationships between entities, or nodes, of the system. Some examples of such systems are the Internet, social networks, and World Wide Web. In the last decade social networks have attracted immense attention in research and industry.

Community detection is a fundamental concept in various fields of science like sociology, biology, computer science, etc. For example, human communities have been studied in social sciences for decades [8, 19]. In biology, for instance, researchers analysed communities in protein interaction networks to find some specific actions in cells [6, 30]. Community detection has also been extensively used in clustering web clients, to provide better services for World Wide Web clients [20].

Community detection approaches can be classified into global and local methods. While global approaches require all information of the entire network, local methods try to find community patterns in subsets of a graph without considering the entire information, resulting in less computation and being more practical, especially when they are applied to large social networks. The main drawback of global methods is that they have to extract pairwise information for all pairs of nodes in the entire graph. Such information might be very expensive to be extracted and impractical for real-world applications. On top of computation expenses, the information of the entire network is not always available, posing another difficulty for global approaches. On the other hand, local community detection is mostly designed based on finding a community surrounding a starting node without exploring the entire network. As couple examples, HITS [18] and PageRank [39] are popular ranking algorithms which can be seen as local community detections in the network of the web.

This paper is the extension of our previous research [29] in which we briefly introduced a framework for approximating derivatives in graphs and then we proposed the derivate-based community detection (DCD). The method was inspired by geometric active contours [5], an object detection algorithm extensively used in the field of computer vision [12, 13]. The analogy between the discovery of shapes in images and the detection of communities in graphs suggests that an application of the active contours to graph spaces might provide an efficient alternative to existing community detection techniques. In more details, in geometric active contour, an arbitrary curve is evolved until it accurately delineates an object boundary, locations where image intensities change significantly. From this perspective, object boundary can be defined in terms of gradient and curvature, both of which are computed from the derivate of image intensities. The same principle can be translated into graph space provided we can determine the derivatives of a function in graph space.

In this paper, we extend our approach [29] for approximating derivatives in graph space along with mapping few concepts such as gradient and curvature from differential geometry into graph space. In addition, we, also, introduce a novel derivative-based approach based on the concept of surface tension from chemistry in order to track local communities in dynamic networks. We aim to understand and explain communities and their evolution using surface tension, a natural phenomenon which has been comprehensively investigated in chemistry. We know from chemistry that the binding forces between the molecules of a liquid draw the molecules of the substance into a shape that has the least surface area. Putting it differently, a community of similar liquid molecules tends to shape themselves in a way that surface tension is minimised. In an analogues manner, binding forces between nodes of a community inside a network lead to particular patterns for the community.

We modeled surface tension of communities in networks and showed that our model can be used for tracking local communities in networks. We use surface tension as an objective for local communities. To show the surface tension of a community in an acceptable representative of the community's quality, we compared the surface tension of several communities against the conductance, a well-known and widely accepted quality measure for communities [24]. When molecules of the same substance are added to a liquid, the liquid changes its shape so that the surface tension is again minimised. Therefore, surface tension provides a unique ability for tracking local communities in dynamic networks in which new nodes are added over time. In other words, when a node is a candidate of inclusion in a local community, it will be included only if the surface tension of the community is reduced or remains unchanged. Our competitive results on finding local communities using DCD and tracking local communities using surface tension with ground truth datasets show the practicality of the proposed approaches and, more importantly, the usefulness of the concepts derivatives in graph space.

2 Related Work

There are only a few studies on the derivatives in networks. Friedman and Tillich [14] extended some concepts from calculus to networks. They mapped the concepts such as differentiable functions, boundary, and gradient over the graph in order to create a wave equation to investigate the changes in the connectivity of the nodes in a given graph. In another research, Diao et al. [10] 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. In addition, when it comes to finding higher order derivatives (second or higher) Solomon's framework is computationally unfeasible since it needs to solve an exponential 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 exponential approach [34]. The proposed approach also does not deal with the mathematical difficulty and limitations of Friedman and Tillich [14] and Diao et al. [10] approaches. In another study, Van Gennip et al. proposed and derived a graph curvature, analogous to mean curvature in continuous domain. Since the curvature of a vector in continuum is defined as the divergence of normal vector field, the authors first derived the normal of a vertex and then defined the curvature at that vertex by taking the divergence of the normal vector. Their approach is valid for unidirectional graphs and was assumed that no isolated node or self-loop exists. Another study closely related to differentiation over graph space has been done in image processing domain by Ta et al. [37]. They defined the directional derivative of a function at vertex along an edge analogous to continuous domain. Similar to our approach they derived the derivative from a numerical point of view, where it has been approximated by difference function. Although their definition satisfies basic derivative properties, but it only relies on inspiration from continuous. However, our approach to extract derivatives in discrete domain follows up Taylor expansion and satisfies many properties in continuous derivatives like additive and multiplicative properties.

In local community detection, most algorithms try to find a community surrounding a node or a seed. There exist several local community detection methods; however, due to limited space, only the most relevant approaches are discussed. 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 [7, 21]. In this class of algorithm, at each step the candidate node which has the highest quality (based on local modularity) is added to the community until the maximum size of community is reached. Mahoney et al. [25] proposed local spectral clustering (LSP). Spectral clustering uses the eigenvectors of the Laplacian adjacency matrices of graphs as a basis of a clustering algorithm such as hierarchical or K-means in order to cluster vertices into communities [26, 28]. Andersen 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 be trapped in the same community rather than traveling to other communities.

There are two main approaches for tracking communities in dynamic networks: snapshot model and temporal smoothness. 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 [42]. 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 [11] use Girvan–Newman modularity-based community detection for both finding and tracking communities. Tong et al. [38] suggested low rank approximation for detecting dynamic networks; however, their research lacks evaluation. Xu et al. [41] used a hidden Markov model to address dynamic networks. In vertex-centered methods [4], 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. [23] 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 et al. [35]. Graphscope [36] is a parameter-free algorithm which mines time evolving graphs in order to find communities, and their change over time. Nguyen et al. [27] 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 and Hamzeh [31] developed a two-phased model that is comprised of a global and local method. In the first phase, they find global communities and, in the second phase, they find and track local communities in the detected clusters using the global approach. Shang et al. [32] 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.

3 Derivatives in Graph Space

To facilitate the understanding of these concepts in graph space, a few definitions are provided.

Definition 1 (Graph Space) Graph space is the world that defines the graph $G(V, E)$. It consists of a set edges (E), and a set of nodes (V).

Definition 2 (Dimension of a Node in the Graph Space) The degree of a node represents the dimension of the node in the graph space. Any point in Euclidean space has three dimensions, whereas any node in graph space has its own number of dimensions. In Euclidean space, the three dimensions are x, y, and z, whereas in graph space a node with ten neighbours has a dimension of ten and a node with two neighbours has only two dimensions.

Definition 3 (A Shape in Graph) In Euclidean geometry, a shape is an object that is limited by an external boundary, or surface. In graph $G(V, E)$, shape $\chi(\nu, \xi)$

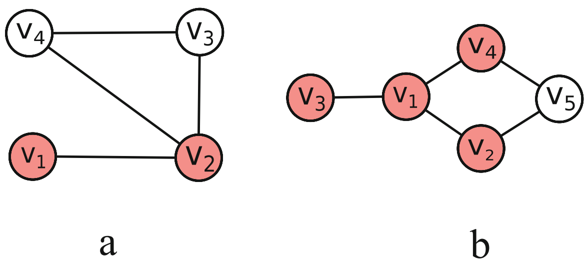

Fig. 1 Examples of shapes in graphs

consists of set of nodes ν that are connected with the edges ξ, $(\nu \subseteq V, \xi \subseteq E)$. A shape can also be seen as a connected subgraph. Each shape in graph space has its own boundary.

Definition 4 (Boundary of a Shape in Graph Space) The boundary of a shape in a graph is the set of nodes that belong to the shape and have common edge(s) with nodes outside the shape, formally a node v_i is on the boundary of shape χ if $\exists e_{ij} \in E | v_i \in \nu \wedge v_j \in V \wedge v_j \notin \nu$. In other words, if a node has a neighbour outside of the shape, it is on the boundary, or the edge, of the shape. Figure 1 demonstrates two shapes in two different graphs. In Fig. 1a, the nodes in red colour compose a shape which consists of only two nodes. Figure 1b shows a shape that is comprised of four nodes. Nodes v_2 and v_4 in Fig. 1b are considered the external boundary of the shape.

Definition 5 (Functions in Graph Space) A function defines a relation between an input set and an output set where each input is related to exactly one output. A function has its domain and codomain which is showed with expression $f : X \to Y$. In Euclidean space, the derivative of function f shows the rate of change of f at a given point in space.

In graph space, derivative of a function shows the rate of change of the function at a given node. More precisely, in a graph, the derivative is defined as the rate of change of function $F(v)$ at a given node v. The set of nodes V should appear in the domain for the functions in the graph space. Codomain varies depending on the definition of the function F. By adding the time dimension, rates of changes can be tracked with respect to two criteria: structure and time. As a result, two partial derivatives can be defined for a given node. For example, for a function F, which returns the degree of a given node, $\frac{\partial F}{\partial v}$ represents the rate of change of the degree with respect to the structure, and $\frac{\partial F}{\partial t}$ represents the rate of change of the degree of a node with respect to time.

Mapping the concepts of derivative to graph space enables us to use varieties of derivative-based tools in the graph space. Graphs are discrete by nature, and like many discretised problems, to extend continuous mathematics to the graphs, numerical analysis tools should be considered. In this section, a novel approach to determine derivatives of function in graph space, which is similar to the finite difference methods, is proposed.

3.1 Discretisation and Finite Difference

Discretisation, a term in numerical analysis which was introduced by Ames [1] in 1965, is the process that converts continuous functions to discrete ones. Continuous functions have a continuous domain. In the discretisation process, the function's domain is reduced to a set of finite values. Analytical solutions for finding derivatives of a given function require the continuity of the function in their domain. Numerical solutions find derivatives by using only discrete points of the domain. That is to say to use numerical solutions, the functions must be either discrete by nature or to be discretised. The task of discretisation and approximating derivatives is called finite difference method.

Finite difference methods provide straightforward ways for finding derivatives and solving differential equations by replacing partial derivatives with suitable algebraic difference quotient. This results into algebraic system of equations. Approximated derivatives are solutions of the systems of equations. Such systems of equations can be easily solved by computers. This explains the rapid growth of finite difference applications in the last few decades. Finite difference methods are used when a space or a function is discrete by nature such as graph space. To briefly explain how finite difference works, an example will be used. Finite difference methods approximate derivatives by using Taylor series [9] in Eq. (1)

$$f(x + \Delta x) = f(x) + (\Delta x)f'(x) + \cdots + \frac{(\Delta x)^i}{i!} f^i(x) + \cdots \tag{1}$$

In Fig. 2a, the goal is to find the derivative, or rate of change, of $f(x)$ at point x. To find the derivative of f at point x using analytical methods, both the equation of f and the value of x are required.

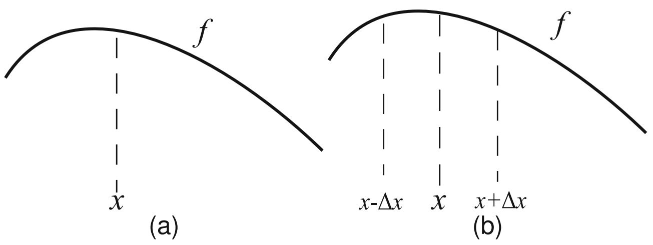

Fig. 2 (**a**) Approximating derivative of f at x, (**b**) discretising f into three points: $x - \Delta x$, x, and $x + \Delta x$

In contrast, numerical methods, first, discretise the domain into finite number of points; then, they approximate the derivative of f at x. The discretisation of f into three points is shown in Fig. 2b.

Since function f is known, the values of $f(x-\Delta x)$, $f(x)$, and $f(x+\Delta x)$ are also known. In many real-world applications, f is not properly defined. For example, it can be assumed that three sensors are located at $x-\Delta x$, x, and $x+\Delta x$. Each sensor reports the temperature of that point, and the goal is to approximate the rate of change of the temperature, or the derivative of temperature, at x using the collected data from sensors and sensors' locations. This means the derivative of temperature can be calculated even though there is no clear definition for temperature's equation.

According to the Taylor series [9], the numerical approximation of the first-order derivative for a function $f(x)$ is

$$f'_{forward}(x) = \frac{f(x+\Delta x) - f(x)}{\Delta x} + O(\Delta x) \tag{2}$$

$O(\Delta x)$ refers to the omitted elements of the Taylor expansion. Similarly, the first-order backward derivative is

$$f'_{backward}(x) = \frac{f(x) - f(x-\Delta x)}{\Delta x} + O(\Delta x) \tag{3}$$

Alternatively, values of f in all $x - \Delta x$, x, and $x + \Delta x$ can be considered for approximating derivative of f at point x:

$$f(x+\Delta x) = f(x) + (\Delta x) f'(x) + \cdots \tag{4}$$

$$f(x-\Delta x) = f(x) - (\Delta x) f'(x) + \cdots \tag{5}$$

By deducting Eq. (5) from Eq. (4), the second-order first derivative can be approximated as follows:

$$f'_{central}(x) = \frac{f(x+\Delta x) - f(x-\Delta x)}{2\Delta x} + O(\Delta x)^2 \tag{6}$$

The terms $O(\Delta x)$ and $O(\Delta x)^2$ in Eq. (2) and Eq. (6) are called truncation error and represent the remaining parts on the right side of Eq. (1) which are neglected if one wishes to *approximate* derivatives. Figure 3 illustrates the approximated solutions for derivative of f at x using first-order backward, first-order forward, and second-order central derivative approximations.

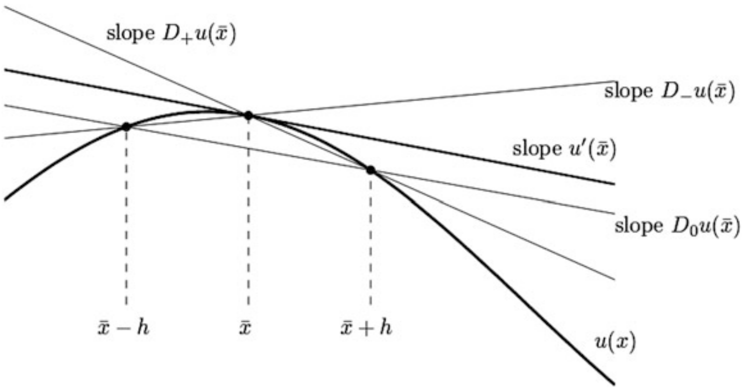

Fig. 3 Approximating the derivative of $f(x)$ using Taylor series

3.2 Approximating Derivatives in Graph Space

Figure 2b, which represents discretisation in Euclidean space, can be extended to graph space. This can be seen in Fig. 4a. The first noticeable difference between the proposed framework here and normal finite difference method is the dissimilarity between Δx and the distances d_1 and d_2 in graph space. While a continuous space can be easily discretised into regular intervals, the interval or distances between different nodes in graph space are not necessarily regular. For instance, the distance between people in a social network can be represented by their profile differences, and, since individuals differ, the distance between individuals is not regular.

By extending Eq. (2) and Eq. (6) to graphs, first-order derivative of F at node v_i is

$$F'_v(v_i) = \frac{F(v_{i+1}) - F(v_i)}{v_{i+1} - v_i} \tag{7}$$

where $v_{i+1} - v_i$ shows the distance, or dissimilarity, between these two nodes and is equal to d_1.

Following the same logic, the second-order first derivative is

$$F'_v(v_i) = \frac{F(v_{i+1}) - 2F(v_i) + F(v_{i-1})}{d_1 + d_2} \tag{8}$$

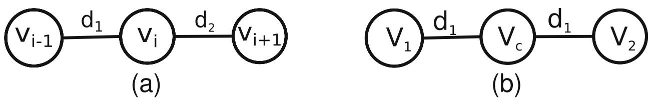

Fig. 4 (**a**) Example graph with three nodes, (**b**) derivatives of f at v_c which has two neighbours

where $d_1 = v_i - v_{i-1}$ and $d_2 = v_{i+1} - v_i$. d_i, in general, show the difference between the nodes in the graph. Applying this model to **weighted graphs** is straightforward. If the weight of the edge that connects v_i to v_{i-1} is w, then $d_i(w) = \dfrac{d_i}{w}$.

The second derivative according to the Taylor series:

$$f(x + \Delta x) = f(x) + \Delta x f'(x) + \frac{(\Delta x)^2}{2!} f''(x) + O(\Delta x)^3 \tag{9}$$

In the rest of this section, after analysing two examples, a general model for finding the derivatives of a given function $F(v)$ is proposed.

Example 5 Finding first and second derivative of F at node v_c with two neighbours (Fig. 4b).

The following equations can be extracted from Taylor expansion:

$$F(v_c + d_1) = F(v_1) = F(v_c) + d_1 F'_v(v_c) + \frac{d_1^2}{2} F''_v(v_c) \tag{10}$$

$$F(v_c + d_2) = F(v_2) = F(v_c) + d_2 F'_v(v_c) + \frac{d_2^2}{2} F''_v(v_c) \tag{11}$$

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

$$\begin{bmatrix} d_1 & \frac{d_1^2}{2} \\ d_2 & \frac{d_2^2}{2} \end{bmatrix} \begin{bmatrix} F'(C) \\ F''(C) \end{bmatrix} = \begin{bmatrix} F(V_1) - F(C) \\ F(V_2) - F(C) \end{bmatrix} \tag{12}$$

In Eq. (10), only three first elements of Taylor expansion Eq. (1) are used. The omitted elements contribute to error of the approximation which will be extensively discussed.

Example 6 The current node v_c (the subscript c stands for the *current*) has three neighbours in Fig. 5a and the goal is to approximate first and second derivatives of F at v_c.

Following equations can be extracted from Fig. 5a by expanding Taylor series up to three elements for each neighbour of v_c:

$$F(v_c + d_1) = F(v_1) = F(v_c) + d_1 F'_v(v_c) + \frac{d_1^2}{2} F''_v(v_c) \tag{13}$$

$$F(v_c + d_2) = F(v_2) = F(v_c) + d_2 F'_v(v_c) + \frac{d_2^2}{2} F''_v(v_c) \tag{14}$$

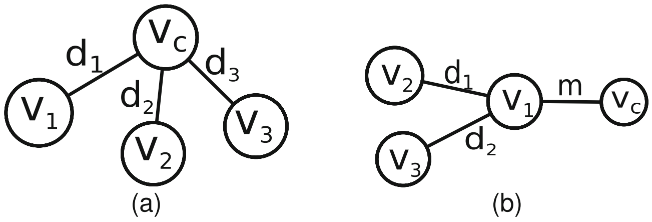

Fig. 5 (**a**) Approximating derivatives of F at v_c which has three neighbours, (**b**) non-central nodes

$$F(v_c + d_3) = F(v_3) = F(v_c) + d_3 F_v'(v_c) + \frac{d_3^2}{2} F_v''(v_c) \tag{15}$$

Accordingly, the system of equations is

$$\begin{bmatrix} d_1 & \frac{d_1^2}{2} \\ d_2 & \frac{d_2^2}{2} \\ d_3 & \frac{d_2^3}{2} \end{bmatrix} \begin{bmatrix} F'(v_c) \\ F''(v_c) \end{bmatrix} = \begin{bmatrix} F(v_1) - F(v_c) \\ F(v_2) - F(v_c) \\ F(v_3) - F(v_c) \end{bmatrix} \tag{16}$$

This is an overdetermined system with three equations and two unknowns. Overdetermined systems are usually inconsistent and have 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. Alternatively, and preferably least squares approximation methods, which are discussed later, can be used for solving overdetermined systems.

Although Example 6 did not need for higher derivatives, at the price of higher computations, by expanding one more element of Taylor series for each neighbour of v_c and adding the third derivatives to the unknowns, the overdetermined system is converted to a determined system.

The resulting system of equations:

$$\begin{bmatrix} d_1 & \frac{d_1^2}{2} & \frac{d_1^3}{3!} \\ d_2 & \frac{d_2^2}{2} & \frac{d_2^3}{3!} \\ d_3 & \frac{d_2^3}{2} & \frac{d_3^3}{3!} \end{bmatrix} \begin{bmatrix} F'(v_c) \\ F''(v_c) \\ F'''(v_c) \end{bmatrix} = \begin{bmatrix} F(v_1) - F(v_c) \\ F(v_2) - F(v_c) \\ F(v_3) - F(v_c) \end{bmatrix} \tag{17}$$

In both Examples 5 and 6, node v_c was located between multiple nodes. A new challenge is posed when derivatives at a node with only one neighbour are desired. Approximating the derivatives of F at v_3 in Fig. 5a is such an example. It will be shown that derivatives of such nodes are also calculable. However, before that two new definitions need to be provided.

Definition 6 (Central Node) A node is called central node, if it has more than one neighbour; nodes v_c in Figs. 4b and 5a are examples of central nodes. This definition has no relation with the degree of centrality.

Definition 7 (Non-central Node) A node is non-central, if it is not located between at least to other nodes. Putting differently, a non-central node has only one neighbour.

Example 7 illustrates the approach for approximating derivative of a function at a non-central node.

Example 7 The goal is to find first, second, and third derivatives of F at the current node v_c in Fig. 5b. Node v_c is a non-central node and has only one neighbour v_1. This example shows how by using Taylor series and values of F at v_2 and v_3 (neighbours of the non-central node's neighbour).

Writing Taylor expansion for node v_1 is straightforward

$$F(v_c + m) = F(v_1) = F(v_c) + mF'_v(v_c) + \frac{m^2}{2}F''_v(v_c) + \frac{m^3}{3!}F'''_v(v_c) \tag{18}$$

A slightly different approach is taken to write Taylor expansions of F at nodes v_2 and v_3. The Taylor expansions of F at v_2 and v_3 are as follows:

$$\begin{aligned} F(v_c + m + d_1) = F(v_2) = {} & F(v_c) + (m + d_1)F'_v(v_c) \\ & + \frac{(m + d_1)^2}{2}F''_v(v_c) + \frac{(m + d_1)^3}{3!}F'''_v(v_c) \end{aligned} \tag{19}$$

$$\begin{aligned} F(v_c + m + d_2) = F(v_3) = {} & F(v_c) + (m + d_2)F'_v(v_c) \\ & + \frac{(m + d_2)^2}{2}F''_v(v_c) + \frac{(m + d_2)^3}{3!}F'''_v(v_c) \end{aligned} \tag{20}$$

Subsequently, first, second, and third derivatives can be approximated by solving the following system of equations:

$$\begin{bmatrix} m & \dfrac{m^2}{2} & \dfrac{m^3}{3!} \\ (m+d_1) & \dfrac{(m+d_1)^2}{2} & \dfrac{(m+d_1)^3}{3!} \\ (m+d_2) & \dfrac{(m+d_2)^2}{2} & \dfrac{(m+d_2)^3}{3!} \end{bmatrix} \begin{bmatrix} F'(v_c) \\ F''(v_c) \\ F'''(v_c) \end{bmatrix} = \begin{bmatrix} F(v_1) - F(v_c) \\ F(v_2) - F(v_c) \\ F(v_3) - F(v_c) \end{bmatrix} \tag{21}$$

A General Framework for Approximating Derivatives of a Function in Graph Space The approximation of the derivatives of a function at a given node in graph space depends on the following factors:

- **Position of the node:** A node can central or non-central (Definitions 6 and 7).
- **Number of neighbours:** For a central node with n neighbours derivatives one to n can be approximated. For a non-central node where its only neighbour has $n-1$ nodes, derivatives one to n can be approximated.
- **Desired order of derivative:** A general framework must answer different users' requirements. In some cases, users may only need up to second derivative, and in some other cases, they may need up to higher derivatives.

Based on the first factor, position of the node, the general framework is broken into two categories. Two remaining factors, number of neighbours and desired order of derivatives, are analysed in each category.

Derivatives at Central Nodes Figure 6a shows a central node v_c that has n neighbours. This means derivatives one to n are available for this node.

Taylor series equation for the ith $(1 \leq i \leq n)$ neighbouring node of v_c is

$$F(v_c + d_i) = F(v_i) = F(v_c) + d_i F_v^1(v_c) + \cdots + \frac{d_i^n}{n!} F_v^n(v_c) \tag{22}$$

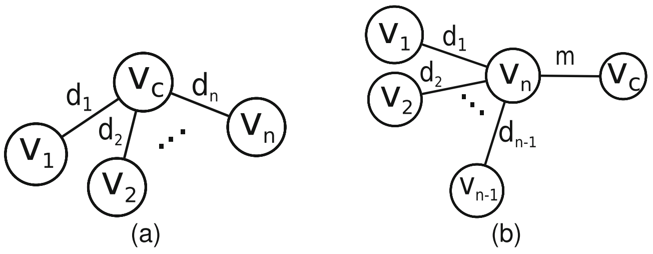

Fig. 6 (**a**) Approximating derivative of F at node v_c with n neighbours, (**b**) approximating derivative of F at the non-central node v_c

These equations result into the following system of equation:

$$\begin{bmatrix} d_1 \dots \dfrac{d_1^n}{n!} \\ \\ \dots \\ \\ d_n \dots \dfrac{d_n^n}{n!} \end{bmatrix} \begin{bmatrix} F^1(v_c) \\ \\ \dots \\ \\ F^n(v_c) \end{bmatrix} = \begin{bmatrix} F(v_1) - F(v_c) \\ \\ \dots \\ \\ F(v_n) - F(v_c) \end{bmatrix} \tag{23}$$

Equation (23) is a system of linear equations with n equations and n unknowns. This system of equations calculates first to nth derivatives of F at node v_c. However, in some applications, the higher orders of derivatives are not necessary. For example, determining the curvature of shape at given node in graph space requires only first and second derivatives. In other words, some applications only need up to mth derivative ($1 \leq m \leq n$). In such cases, approximating $n - m$ extra unknowns is unnecessary. Considering extra unknowns becomes particularly challenging or computationally expensive when m is a large number. In such cases, the number of unknowns is reduced to m. This can be done by modifying Eq. (22) so that it incorporates only m elements in expansion of Taylor series for each neighbouring node. This resulting equation is

$$F(v_c + d_i) = F(v_i) = F(v_c) + d_i F_v^1(v_c) + \dots + \frac{d_n^m}{m!} F_v^m(v_c) \tag{24}$$

where i ($1 \leq i \leq n$) represents an equation for each neighbour of v_c, and m ($1 \leq m \leq n$) is a constant that represents the desired order of derivatives. Equation (24) represents n equations where each equation has j unknowns. The unknowns are determined by solving the following overdetermined systems of equations:

$$\begin{bmatrix} d_1 \dots \dfrac{d_1^m}{m!} \\ \\ \dots \\ \\ d_n \dots \dfrac{d_n^m}{m!} \end{bmatrix} \begin{bmatrix} F^1(v_c) \\ \\ \dots \\ \\ F^m(v_c) \end{bmatrix} = \begin{bmatrix} F(v_1) - F(v_c) \\ \\ \dots \\ \\ F(v_n) - F(v_c) \end{bmatrix} \tag{25}$$

In Eq. (25), the number of unknowns is less than number of equations. In such cases, the least square approximation method is used to find the answers of Eq. (25). Reduced QR factorisation [16] and singular value decomposition (SVD) [22] are two well-known methods for approximating the least square solutions. While SVD method is more accurate, QR method is faster.

In general terms of linear algebra, a system of equations is expressed as $Af = b$. A system has no solution if the determinant of A is equal to zero. Considering the constraint matrix in Eq. (23) or Eq. (25), the determinant is zero when there exist i and j, ($1 \leq i \leq n$), ($1 \leq j \leq n$), and $i \neq j$. In other words, there are i and j

in the first matrix of Eq. (23) so that $d_i = d_j$. That means node v_c has exactly the same distance to two of its neighbours v_i and v_j. Putting it differently, v_i and v_j are equivalent to v_c. For instance, in social network context, this implies that the difference between profiles of v_c and v_i is exactly equal to profile difference of v_c and v_j. In case of such occurrences, the approach here is to alternatively omit v_i and v_j to make system of equation solvable. If the difference between two alternative approximations is more than a given threshold (i.e., noticeable), then a new meta-node v_x is created and replaces both v_i and v_j, and $F(v_x) = \frac{F(v_i)+F(v_j)}{2}$.

Derivatives at Non-central Nodes Figure 6b shows one of the peripheral nodes as the current node v_c for which the derivative is approximated. The neighbour of the current 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 $F(v_c)$ are approximated by solving the following system of equations:

$$F(v_c + m) = F(v_n) = F(v_c) + mF'_v(v_c) + \cdots + \frac{m^n}{n!}F^n_v(v_c) \tag{26}$$

All other nodes v_i where $1 \le i \le n-1$ have the following equation:

$$\begin{aligned} F(v_c + m + d_i) = F(v_i) = F(v_c) + (m + d_i)F'_v(v_c) + \cdots \\ + \frac{(m + d_i)^n}{n!}F^n_v(v_c) \end{aligned} \tag{27}$$

4 Community Detection Using Derivatives

4.1 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 [5]. A related approach, based on differential geometry, was devised in 1997. Due to its efficiency, autonomy, and unsupervised nature geometric active contours [5] is used extensively for detecting object 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 Eq. (28), describes how fast the curve changes its tangent or direction

$$\kappa = \frac{f''(x)}{(1 + f'^2(x))^{\frac{3}{2}}} \tag{28}$$

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

$$\nabla = \frac{\partial}{\partial x}i + \frac{\partial}{\partial y}j + \frac{\partial}{\partial z}k \tag{29}$$

Assuming three-dimensional Euclidean space, the gradient of $f(x, y, z)$ is obtained by applying the vector operator ∇ to the scalar function $f(x, y, z)$ as defined in Eq. (30)

$$\nabla f(x, y, z) = \frac{\partial f}{\partial x}i + \frac{\partial f}{\partial y}j + \frac{\partial f}{\partial z}k \tag{30}$$

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 at that direction. In images, gradient is obtained by subtraction of gray 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 Eq. (31). The velocity decides the likelihood of a pixel being included in the shape

$$s = g\kappa \boldsymbol{n} - (\nabla g\boldsymbol{n})\boldsymbol{n}, \text{ where } g = \frac{1}{1 + |\nabla I|^2} \tag{31}$$

where $\boldsymbol{n}$ denotes the normal direction and I the pixel values in an image with $|\nabla I|$ as the magnitude of the gradient between two pixels [5]. 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.

4.2 Finding Local Communities

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.

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

$$F(v_i, v_j) = 1 - \frac{|N(v_i) \cap N(v_j)|}{|N(v_i) \cup N(v_j)|} \tag{32}$$

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 initialise community in Algorithm 1. The shape χ 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 Eq. (33)

$$s(v_i, v_p) = \frac{\kappa(v_i, v_p)}{1 + \alpha|\nabla F(v_i, v_p)|^2} - \arctan(|F'(v_i, v_p)|) \tag{33}$$

In Eq. (33), the curvature is represented by $\kappa(v_i, v_p)$, which is defined in Eq. (34). The magnitude of the gradient $|\nabla F(v_i, v_p)|$ describes the difference between v_i and the candidate node v_p. The parameter α moderates the difference between nodes. The larger the 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

$$\kappa(v_i, v_p) = \frac{F''(v_i, v_p)}{\left(1 + \left(F'(v_i, v_p)\right)^2\right)^{\frac{3}{2}}} \tag{34}$$

As shown in Eq. (34), curvature uses 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 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 Fig. 5, values of curvature and gradient for two simple graphs are shown. Using the Eq. (33), the velocity from v_5 towards v_p is -0.06 in Fig. 5a; thus, v_p will not be included in the community. In Fig. 5b, 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 (Fig. 7).

Algorithm 1 Derivative-based community detection

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

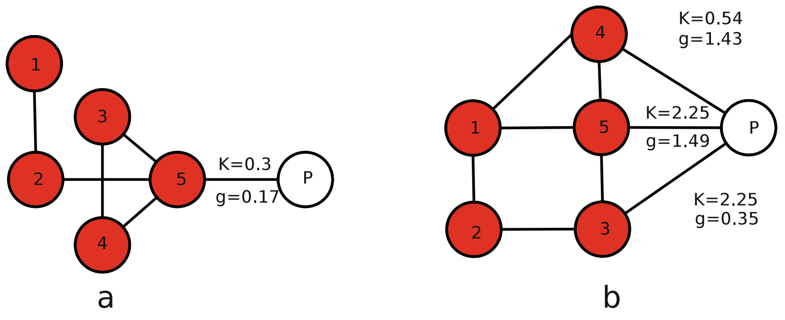

Fig. 7 In both (**a**) and (**b**), shape χ consists of v_1, v_2, v_3, v_4, and v_5 and v_p is the candidate for the inclusion. In (**a**), values of curvature and gradient from v_5 towards v_p are shown. In (**b**), values of curvature and gradient from v_3, v_4, and v_5 towards v_p are shown

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 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 8 shows an example of the proposed algorithm.

In Eq. (33), the velocity function has only one parameter, α. To give the user control over size and quality of the desired communities α is added to the inclusion criteria. The larger the α is, the stronger the effect of the gradient, and therefore the sharper the edge.

5 Tracking Local Communities Using Surface Tension

To track communities, we use structural similarity defined in Eq. (32). The structural similarity shows the proportion of the common neighbours. In investigating local communities, a node has one of the following situations: outside of the boundary of a community, on the boundary of a community, or inside a community (without any neighbours in outside). This is illustrated in Fig. 9. As it is shown in Fig. 9, two binding forces are affecting a node on the boundary. We simulated the inside and outside pressures on the surface of a community using these pressures (binding forces). $P_{outside}$ and P_{inside} are modeled by structural similarity of the nodes on the boundary of the community to the nodes inside and outside of the community

$$K = \sum_{i=1}^{n} \kappa(v_i, C) \tag{35}$$

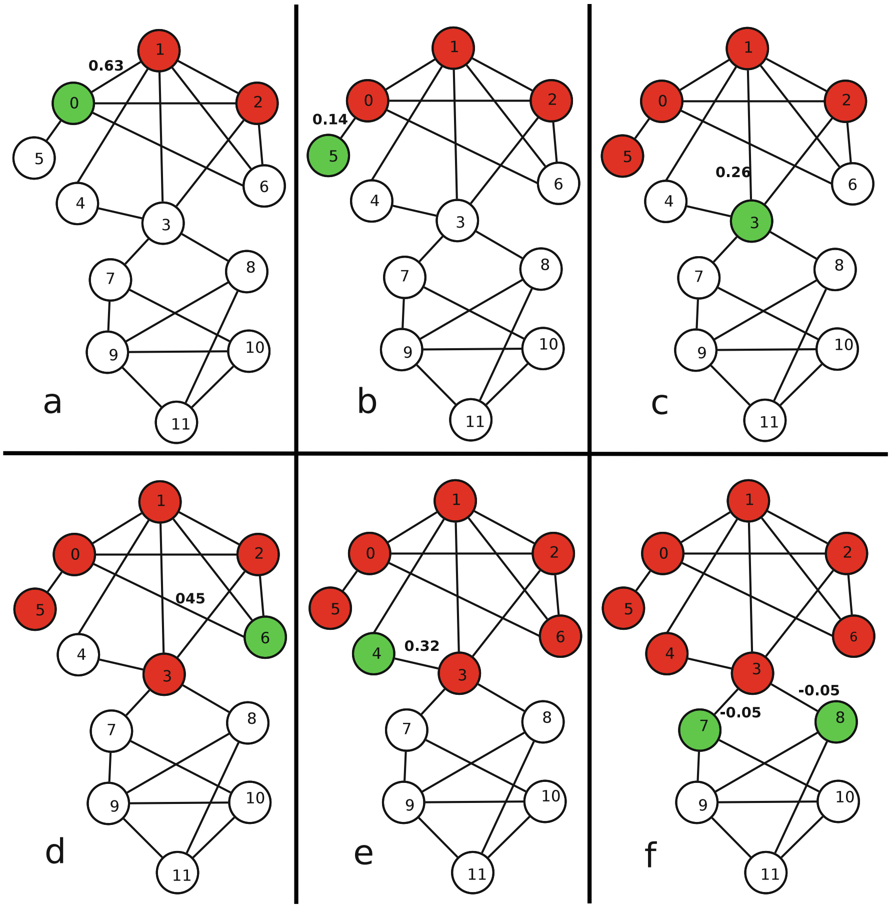

Fig. 8 The red nodes show the current community and the green nodes are candidates for inclusion. The number on the edges shows the velocity from a node to the candidate. When the velocity towards all neighbouring nodes is negative, the algorithm stops

$$P_{outside} = \sum_{i=1}^{n} \sum_{j=1}^{m} similarity(v_i, outside\_neighbour_j(v_i)) \tag{36}$$

$$P_{intside} = \sum_{i=1}^{n} \sum_{j=1}^{m} similarity(v_i, inside\_neighbour_j(v_i)) \tag{37}$$

where n is the number of nodes on the surface of a community and m represents the number of inside or outside neighbours for the ith node on the surface. In our model, we use the radius of curvature towards inside the community. Thus, the surface tension of a community can be represented in Eq. (38).

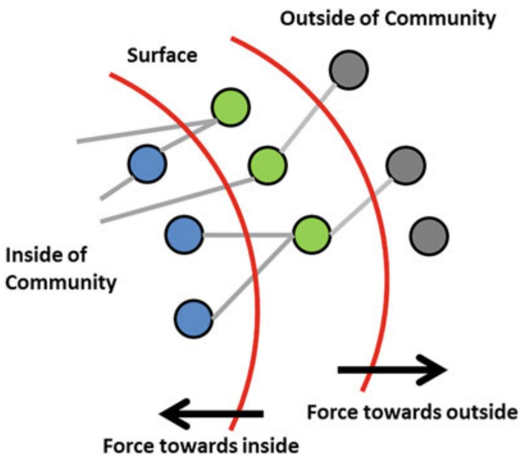

Fig. 9 Surface of a community and its binding forces

$$\gamma = (P_{outside} - P_{inside})K \tag{38}$$

where κ was defined in (34). Substances are shaped in a way that the tension on their surface in minimised. Following a similar logic, a node is added to a community if surface tension of the community is reduced or it remained unchanged. Our method is able to track local communities with temporal smoothness changes in a network. In temporal smoothness, there is a stream of atomic changes. The community updates itself triggered by a change. The criteria for adding a new node to community is

$$\gamma_{new} - \gamma_{old} \leq \alpha \tag{39}$$

α, which is a non-negative value, is the tolerance threshold. Small values of α allow inclusion of nodes which may slightly increase community's surface tension and, therefore, community's quality. Our experiments show $\alpha = 0$ is a very strict criteria and does not allow inclusion of the nodes which their impact on worsening the quality of community is negligible and close to zero. Because of the tolerance threshold, some nodes may decrease the community's quality, but the quality is expected to increase again when new nodes are added. In other words, exclusion of the nodes that may slightly decrease the quality (or increase the surface tension) prevents the inclusion of some nodes which can increase the quality considerably.

As stated in Eq. (38), to track a community of three vectors keep curvature of boundary nodes towards community, similarity to outside neighbours, and similarity to inside neighbours. One approach is to recalculate the surface tension whenever a new node, based on Eq. (39), is added. However, in a more efficient approach, once a new node is added to boundary, new values for the necessary areas of the three mentioned vectors need to be recalculated.

6 Experimental Evaluation

6.1 Community Detection

Comparing the outcome of local spectral clustering (LSP) [25] 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 et al. [17]. The parameter α in DCD defines the stringency of the inclusion criterion, with larger values being more restrictive. Unlike LSP, DCD stops when according to Eq. (33), no further candidate nodes qualify for inclusion.

Some of the most widely known measures for determining the quality of local communities are intra-cluster density, relative density, and conductance. Intra-cluster density is the fraction of the number of edges inside the community to total number of edges in network. 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. Conductance is defined as

$$\mathit{Conductance(C)} = \frac{vol(C, \bar{C})}{min\Big(vol(C, G), vol(\bar{C}, G)\Big)} \tag{40}$$

In Eq. (40), 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 $vol(C_1, C_2) = \sum_{i \in C_1} \sum_{j \in C_2} A_{ij}$, where A is the adjacency matrix. *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 [17, 24, 25], we choose conductance as a standard quality measure.

The graphs used in the experimentation are Facebook graph FB-JHK of John Hopkins University with 5180 nodes and 186,572 edges, and FB-CALTC of California Institute of Technology with 769 nodes and 33,312 edges, both captured in September 2005 and are part of the FACEBOOK100 dataset [40].

In Fig. 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. Local geodesic spreading (LGS) [3], 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 α, 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

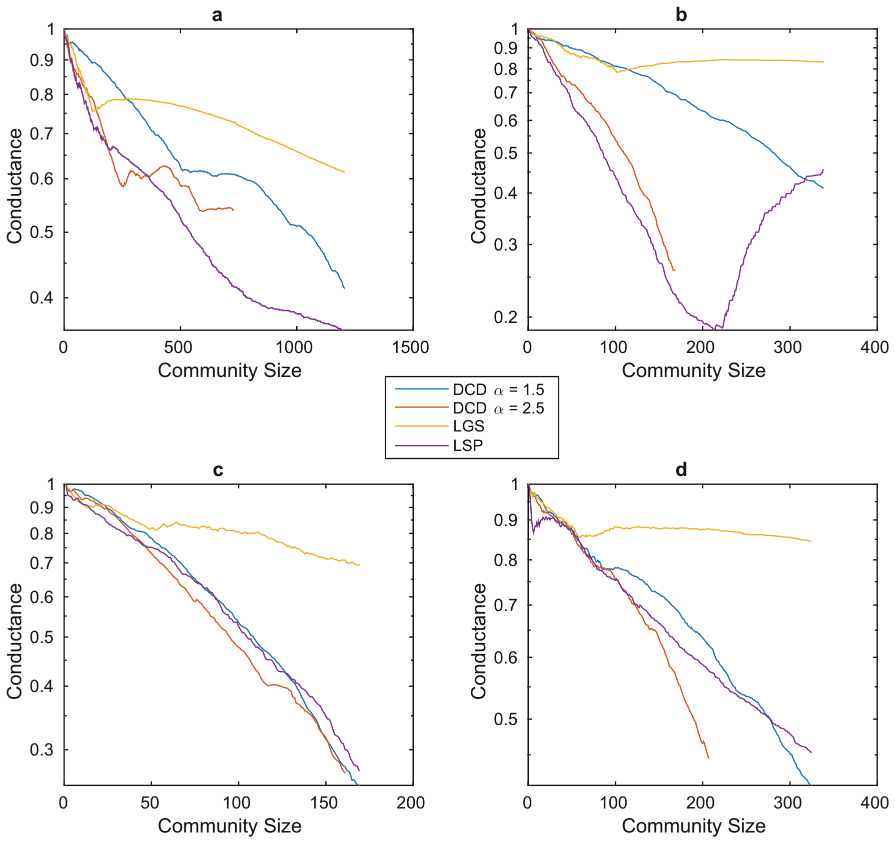

Fig. 10 Conductance plot for different methods and different starting nodes in FB-JHK. Initial seed: (**a**) 2645, (**b**) 3229, (**c**) 3554, and (**d**) 3718

nodes. In Fig. 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 Fig. 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 Fig. 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 Fig. 10 illustrate the difference in performance of DCD that the parameter α entails. This raises the question how to identify the best setting for α. Further investigations, illustrated in Fig. 11a, show that smaller values of α lead to the detection of larger communities, while larger values of α discover small-size communities. Because larger values of α restrict the inclusion of new nodes earlier, the algorithm stops at a smaller community size. Table 1 shows the average sizes of the communities

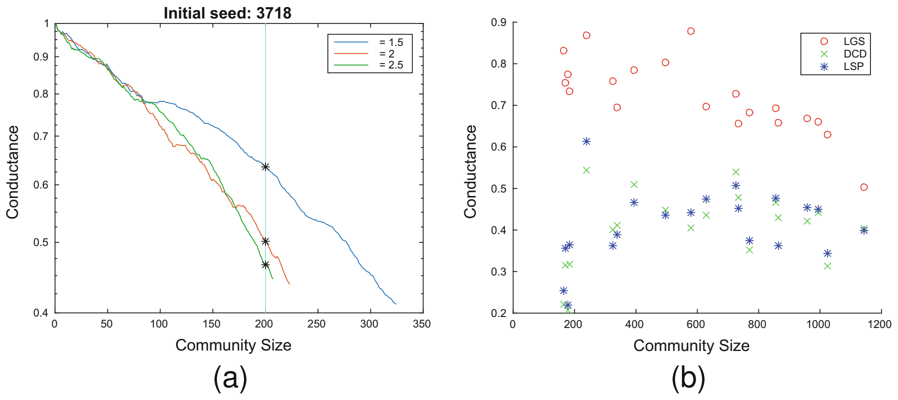

Fig. 11 (**a**) Effect of α on finding local communities around a seed node in FB-JHK, (**b**) conductance of the different detected communities in FB-JHK where $\alpha = 1.5$

Table 1 Effect of α on the size of detected communities in FB-JHK for 20 different initial random seeds

α	1.5	2	2.5
Average size	588.2	242.8	173.1

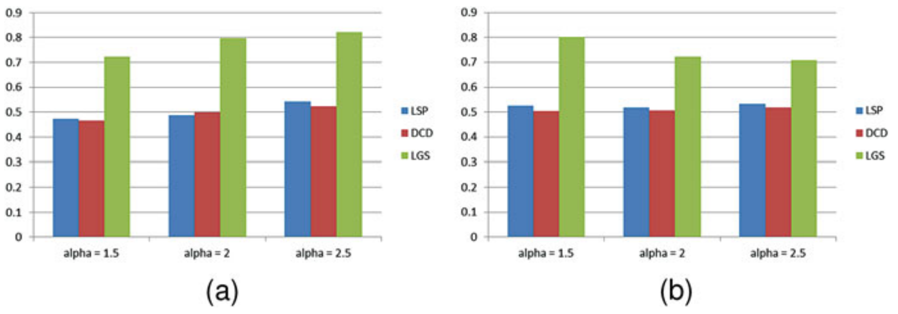

Fig. 12 (**a**) Average conductances of the communities in FB-JHK, (**b**) average conductances of the communities in FB-CALTC

detected with different values of α. 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.

Figure 12a, b 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 α 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 was 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. Figure 12a, b show the conductance of the detected communities for DCD, LSP, and LGS for the 20 random seeds in 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.

6.2 Community Tracking

Community tracking evaluation has two sections. In the first section, we will show why surface tension of a community represents its quality by comparing it to conductance, a very well-known and widely accepted quality measure for communities. In the second section, the effectiveness of the surface tension as local community tracking tool is demonstrated.

6.2.1 Analysing Surface Tension of Communities

To demonstrate the potentiality of surface tension as a local objective or quality measure, we compared it with the conductance for more than 200 communities. These communities were detected by some well-known global and local methods on different networks. All networks in this section are part of FACEBOOK100 dataset [40].

We show the correlation between surface tension of a community and its conductance for several communities in different networks. To find communities, we applied one of the best known global community detection methods, which is proposed by Sobolevsky et al. [33], to FB-Caltech, FB-Trinity, FB-Yale, and FB-Simmon, and then found the correlation between surface tension and conductance of the detected communities. The specification of the mentioned networks is presented in Table 2 and the correlations between surface tension and conductance are presented in Table 3. In another experiment, we calculated the correlation of conductance and surface tension of communities for 100 local communities in FB-UCF and another 100 communities in FB-DUKE. We used local spectral method [25] with different random initial seed for finding these 200 communities in these two networks. The specifications of the networks can be seen in Table 2.

Considering the fact that surface tension is a local concept and only uses the local information of a community, whereas conductance is a global notion and

Table 2 Datasets' details

Networks	FB-Caltech	FB-Trinity	FB-Yale	FB-Simmon
Nodes	669	2613	8578	1518
Edges	33,253	111,996	405,450	65,976
Average degree	43.39	85.72	94.5	86.92

Table 3 The correlation between surface tension and conductance of detected communities by Sobolevsky et al. [33] method

Networks	FB-Caltech	FB-Trinity	FB-Yale	FB-Simmon
Number of communities	10	6	7	6
Correlation	0.7311	0.8768	0.7635	0.8404

Table 4 The correlation between surface tension and conductance of detected communities by local spectral method [25] method.

Networks	FB-UCFA	FB-DUKE
Number of communities	100	100
Correlation	0.9465	0.9286

needs network's entire information, the high correlation between them suggests that surface tension can be seen as a local quality objective (Table 4).

6.2.2 Tracking Local Communities

To evaluate our model for tracking communities, the dynamic community network generator by Görke et al. [15] is used. The benefit of their clustered network generator is its capability to create communities in a dynamic network with an atomic change stream where ground truth is known. 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. We compared surface tension model against the ground truth data. In this experiment, several networks with 1000 nodes and five communities with different intra-cluster and inter-cluster edge probabilities are generated. More intra-cluster and less the inter-cluster probabilities lead to higher quality communities. In the next step, 200 nodes are added to the network through a stream of atomic changes. Our model 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.

To test the performance of our model 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 5.

The precision, recall, and F1 scores for each of the experiments are shown in Fig. 13. 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, it performs better.

Table 5 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

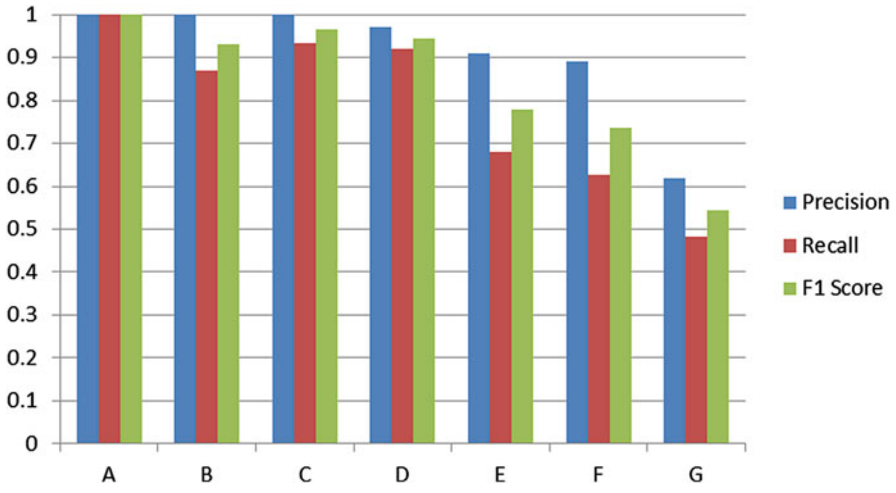

Fig. 13 Precision, recall, and F1 score for each scenario in Table 5

7 Conclusion and Future Works

In this study we have extended the definition of derivatives to graph and approximated derivatives over graph domain. Inspired by geometric active contours, we proposed a method (DCD) that has shown comparable performance to a well-known local community detection algorithm (LSP [25]). While both methods have similar computational complexity, DCD offers more desirable stopping criteria, where unlike LSP it will stop automatically once all qualified nodes have been included in the community. Moreover, we introduced the concept of surface tension, a natural phenomenon which is heavily investigated in chemistry, into networks. According to chemistry, the binding forces between the molecules of a liquid draw the molecules of the substance into a shape that has the least surface area. That is to say, a community of similar liquid molecules tends to shape themselves in a way that surface tension is minimised. Likewise, the binding forces between nodes of a community inside a network lead to particular patterns for a community. A pattern or shape in which the surface tension of community is minimised. We used surface tension as an objective for tracking local communities in dynamic networks. Surface tension provides a unique ability for tracking local communities in dynamic networks in which new nodes are added over time. In other words, when a node is a candidate of inclusion in a local community, it will be included only if the surface tension of the community is reduced or remains unchanged. Our experiments show the effectiveness of the proposed approaches to find and track communities as well as the proposed framework for finding derivatives in graph space.

References

1. Ames, W.F.: Nonlinear Partial Differential Equations in Engineering, vol. 18. Academic, New York (1965)
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, New York (2006)
3. Bagrow, J.P., Bollt, E.M.: Local method for detecting communities. Phys. Rev. E **72**(4), 046108 (2005)
4. 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, Berlin (2016)
5. Caselles, V., Kimmel, R., Sapiro, G.: Geodesic active contours. Int. J. Comput. Vis. **22**(1), 61–79 (1997)
6. Chen, J., Yuan, B.: Detecting functional modules in the yeast protein–protein interaction network. Bioinformatics **22**(18), 2283–2290 (2006)
7. Clauset, A.: Finding local community structure in networks. Phys. Rev. E **72**(2), 026132 (2005)
8. Coleman, J.S.: Introduction to Mathematical Sociology. London Free Press, Glencoe (1964)
9. Courant, R., Friedrichs, K., Lewy, H.: Über die partiellen differenzengleichungen der mathematischen physik. Math. Ann. **100**(1), 32–74 (1928)
10. Diao, P., Guillot, D., Khare, A., Rajaratnam, B.: Differential calculus on graphon space. J. Comb. Theory Ser. A **133**, 183–227 (2015)
11. Falkowski, T.: Community analysis in dynamic social networks. Ph.D. thesis, Otto-von-Guericke-University, Magdeburg (2009)
12. 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, Piscataway (2016)
13. 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)
14. Friedman, J., Tillich, J.P.: Calculus on graphs. arXiv preprint cs/0408028 (2004)
15. Görke, R., Kluge, R., Schumm, A., Staudt, C., Wagner, D.: An efficient generator for clustered dynamic random networks. In: Proceedings of the Design and Analysis of Algorithms – First Mediterranean Conference on Algorithms. Lecture Notes in Computer Science, vol. 7659, pp. 219–233. Springer, Berlin (2012)
16. Golub, G.H., Van Loan, C.F.: Matrix Computations, vol. 3. JHU Press, Baltimore (2012)
17. 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)
18. Kleinberg, J.M.: Authoritative sources in a hyperlinked environment. J. ACM **46**(5), 604–632 (1999)
19. Kottak, C.P.: Cultural Anthropology: Appreciating Cultural Diversity. McGraw-Hill, New York (2011)
20. Krishnamurthy, B., Wang, J.: On network-aware clustering of web clients. ACM SIGCOMM Comput. Commun. Rev. **30**(4), 97–110 (2000)
21. Lancichinetti, A., Fortunato, S., Kertész, J.: Detecting the overlapping and hierarchical community structure in complex networks. New J. Phys. **11**(3), 033015 (2009)

22. Lawson, C.L., Hanson, R.J.: Solving Least Squares Problems, vol. 15. SIAM, Philadelphia (1995)
23. 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, New York (2005)
24. 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)
25. 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)
26. Ng, A.Y., Jordan, M.I., Weiss, Y.: On spectral clustering: analysis and an algorithm. Adv. Neural Inf. Process. Syst. **2**, 849–856 (2002)
27. 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, New York (2011)
28. Rand, W.M.: Objective criteria for the evaluation of clustering methods. J. Am. Stat. Assoc. **66**(336), 846–850 (1971)
29. Rigi, M.A., Moser, I., Rigi, S., Liu, C.: Re-imaging 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 (ASONAM ’17), pp. 974–981. ACM, New York (2017)
30. Rives, A.W., Galitski, T.: Modular organization of cellular networks. Proc. Natl. Acad. Sci. **100**(3), 1128–1133 (2003)
31. Samie, M.E., Hamzeh, A.: Community detection in dynamic social networks: a local evolutionary approach. J. Inf. Sci. **43**(5), 615–634 (2017)
32. Shang, J., Liu, L., Li, X., Xie, F., Wu, C.: Targeted revision: a learning-based approach for incremental community detection in dynamic networks. Phys A: Stat. Mech. Appl. **443**(Suppl. C), 70–85 (2016)
33. Sobolevsky, S., Campari, R., Belyi, A., Ratti, C.: General optimization technique for high-quality community detection in complex networks. Phys. Rev. E **90**(1), 012811 (2014)
34. Solomon, J.: PDE approaches to graph analysis. arXiv preprint, arXiv:1505.00185 (2015)
35. Spiliopoulou, M., Ntoutsi, 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, New York (2006)
36. 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, New York (2007)
37. Ta, V.-T., Elmoataz, A., Lézoray, O.: Partial difference equations over graphs: morphological processing of arbitrary discrete data. In: European Conference on Computer Vision, pp. 668–680. Springer, Berlin (2008)
38. 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, New York (2008)
39. 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)
40. Traud, A.L., Mucha, P.J., Porter, M.A.: Social structure of Facebook networks. Phys. A: Stat. Mech. Appl. **391**(16), 4165–4180 (2012)
41. Xu, T., Zhang, Z., Yu, P.S., Long, B.: Generative models for evolutionary clustering. ACM Trans. Knowl. Discov. Data **6**(2), 7 (2012)
42. 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)