Lecture Notes in Social Networks

Panagiotis Karampelas
Jalal Kawash
Tansel Özyer *Editors*

From Security to Community Detection in Social Networking Platforms

Springer

Lecture Notes in Social Networks

Panagiotis Karampelas • Jalal Kawash
Tansel Özyer
Editors

From Security to Community Detection in Social Networking Platforms

Springer

Editors
Panagiotis Karampelas
Department of Informatics & Computers
Hellenic Air Force Academy
Dekelia, Greece

Jalal Kawash
Department of Computer Science
University of Calgary
Calgary, AB, Canada

Tansel Özyer
Department of Computer Engineering
TOBB University of Economics
and Technology
Ankara, Turkey

ISSN 2190-5428 ISSN 2190-5436 (electronic)
Lecture Notes in Social Networks
ISBN 978-3-030-11285-1 ISBN 978-3-030-11286-8 (eBook)
https://doi.org/10.1007/978-3-030-11286-8

Library of Congress Control Number: 2019935126

Preface

Introduction

This volume is a compilation of the best papers presented at the IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM-2017), held in Sydney, Australia, August 2017. The authors of the selected papers were kindly asked to provide extended versions of the papers that were then subjected to an additional refereeing process. Within the broader context of online social networks, the volume focuses on important and state-of-the-art work in the area of detection and prediction techniques using information found generally in graphs and particularly in social networks.

From Security to Community Detection in Social Networking Platforms

Social networks have not only been under the light due to the vast participation of users but also because they are a very reliable source of information either for collecting security-related information, such as malicious IP addresses, or detecting online communities in diverse contexts using innovative techniques, such as genetic algorithms or surface tension analysis. Additionally, the absence of authentic data due to privacy concerns is addressed. This absence complicates testing novel methodologies in various areas such as financial analysis or mining of unstructured data. Two other chapters present solutions for creating synthetic though realistic data for the aforementioned cases. The rest of the chapters presented in this book propose innovative methods to explore graphs and social networks in an attempt to provide timely, reliable, and useful results in a continuously increasing data-intensive environment. The variety of approaches adopted in the research presented in the book demonstrates the diversity of the application contexts and can act as a source for further research not only in the designated areas presented but in other areas of application.

The first chapter examines how social network analysis can be applied in a complex environment where numerous actors are involved in preserving biodiversity in a protected area. By applying social networking metrics in two ego networks, the authors study the role of the management actors for the preservation of the area at stake and how these metrics can assist in improving the cooperation of environmental conservation in Natura 2000 areas.

In the second chapter, the authors propose a novel methodology to detect social network communities without estimating the number of communities beforehand using a modified genetic algorithm. The reported results have shown that the proposed methodology performs very well even in large datasets.

In the third chapter, the authors recognizing the proliferation of the multidimensionality of current social networks propose a novel Multidimensional Communities Detection Algorithm that is capable of handling outliers and at the same time is able to detect multidimensional communities. The proposed technique can also automatically unfold the hidden communities in a multidimensional context, by creating a novel propagation rule that exploits the most frequently used interaction dimensions among neighbors as an additional constraint for membership selections.

The fourth chapter proposes two methods for detecting local communities. The first one proposes finding derivatives in graph space and, as a result, takes advantage of derivative-based methods into graph theory. The second is inspired by the active contour algorithm in computer vision and explores the concepts of curvature and gradient of the community's boundary. The proposed methods are enhanced by applying the principles of surface tension from chemistry in dynamic networks by adding new nodes. The experiments presented provide promising results regarding the performance of the proposed methods.

In the fifth chapter, the authors propose a new graph embedding approach for attributed graph clustering since nowadays rich and heterogeneous attribute information has become widely available especially in social networks user profiles. By applying the proposed methodology, the authors demonstrate that it is possible to transform the challenging attributed graph clustering problem into a multidimensional data clustering problem. This transformation outperforms traditional attributed graph clustering techniques in terms of effectiveness and efficiency.

The sixth chapter elaborates on the problem of big graph analytics in dynamic graphs. Traditional methods rely on tracking the added or the removed nodes in a graph. The proposed technique takes advantage of additional information that is available by creating the Edge Sample and by employing the discard algorithm, which generates an unbiased estimate of the total number of triangles that may need to be updated due to dynamic changes. The proposed method is evaluated against traditional methods providing promising results.

In the seventh chapter, the authors focus on the problem of semi-structured and structured data that are used in decision-making. Even though there are several data quality management approaches, it is not always feasible to compare or assess the performance of the specific approaches since there are no public datasets available to be used for such purpose. The chapter addresses the specific challenge by proposing a system that is able to produce synthetic semi-structured and structured data satisfying a set of integrity constraints to be used for the assessment of the data quality management methods.

In the eighth chapter, the authors propose a novel methodology for randomly generating entire financial systems while diagnosing the absence of real financial trade datasets for analyzing the impact of financial regulation concerning the collateralization of derivative trades. Based on a novel open-source risk engine, the authors enable data scientists to apply diverse techniques such as data mining, anomaly detection, and visualization to run simulations.

The ninth chapter identifies the need for mining unstructured information, such as in hackers' forums, and proposes a novel methodology and a corresponding tool based on matrix decomposition method to extract latent features of the behavioral information of the users. These features are then used along with some keywords from any language to classify malicious IP addresses found in the forums. Based on the experimental analysis, the authors are able to detect up to three times more malicious IP address than the blacklist of Virus Total.

The tenth and final chapter proposes a method for detecting topic changes between different time periods. The proposed method is based on two techniques: one is from an information-theoretic analysis of the terms distributions, and the second is based on document clustering in periods under review. The validity of the proposed method is tested against various Twitter datasets.

Dekelia, Greece	Panagiotis Karampelas
Calgary, AB, Canada	Jalal Kawash
Ankara, Turkey	Tansel Özyer

Contents

Real-World Application of Ego-Network Analysis to Evaluate Environmental Management Structures

Andreea Nita, Steluta Manolache, Cristiana M. Ciocanea, and Laurentiu Rozylowicz

Abstract In a world in a constant need for development, preserving biodiversity is a daunting task for both governments and NGOs. The centerpiece of successful biodiversity conservation is ensuring cooperation among countless actors involved in the management of protected areas. Social network analysis is a suitable tool for securing essential information for interactions during the management process. To contribute to the debate in the field of governance of protected areas, we illustrate an approach in investigating management of Natura 2000 sites, by considering two real-world management settings in Romania. We evaluate the characteristics of two ego-networks established for the management of two European Union Natura 2000 protected areas in Romania Iron Gates Natural Park administrated by public body owned by state and Lower Siret Floodplain administrated by a regional NGO. The networks were created around administrative bodies of protected area (ego), and include actors directly connected to the ego. After evaluating the most common ego-network metrics that demonstrate the characteristics of each network, we analyzed the strong ties by using Simmelian ties within protected areas management ego-networks and clustered the embedded links in Girvan–Newman groups. The findings suggest that the ego (administrative bodies of protected area) has a critical role in bridging other management actors. The paper tries to identify models of management control by comparing two ego-networks and showing how well connected the administrative bodies of protected areas are. The study provides insights regarding several means to improve the cooperation of environmental conservation in Natura 2000 areas.

Keywords Governance · Natura 2000 network · Simmelian ties · Clusters · Ego-networks · Romania

A. Nita (✉) · S. Manolache · C. M. Ciocanea · L. Rozylowicz
University of Bucharest, Center for Environmental Research and Impact Studies, Bucharest, Romania
e-mail: andreea.nita@cc.unibuc.ro

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_1

1 Introduction

Natura 2000 was established at European Union level as a network of protected areas which includes a representative sample of wildlife and natural habitats of community interest. The selection, designation, and management of Natura 2000 sites are governed by two European Union Directives: Habitats Directive for Sites of Community Interest and Birds Directive for Special Protection Areas [1, 2]. The expansion of Natura 2000 network in EU countries is nearly complete; however, obstacles remain in the implementation and enforcement of the both directives, mostly due to the lack of streamlined management models [3]. The management of these protected sites must consider that a Natura 2000 site is primarily a tool for conservation of social–ecological systems and not focused solely on strictly ecological protection. Given that it is such a complex social–ecological arena, collaborative partnerships may be the key to succeed in implementing conservation goals [4, 5].

Most of the scientific literature focused on Natura 2000 analyzes environmental issues without considering co-management as a key strategy to improve environmental protection and conservation [6, 7]. Considering that Natura 2000 should be focused on human–nature relationships [8], there is the necessity to investigate means to improve and support the effective management of these sites [9], including the investigation of different models of collaboration in the management processes.

While social network literature is extensive regarding global properties of a network [10], relatively few studies discuss the importance of this field in correlating the benefits and constrains within a governance arena in a complex framework [11, 12], such as Natura 2000 [13]. Identifying the actors that are involved in management networks by analyzing network properties at actors level (such as degree and betweenness centrality metrics) may inform about key players; however, the validity of conclusions depends on the quality of data [14]. Management networks are also analyzed at network-level metrics such as network density and clustering informing about interactions of organizations in the entire network [15]. Compared to a full network, analyzing an ego-network presumes the investigation of the personal network, namely, the first order zone or first neighborhood [16]. This analysis focused on the investigation of a particular management actor (ego) and all other organizations (alters) connected to the ego can reveal the importance of the collaborative management of protected areas.

Simmelian ties are usually used to extract the interaction structure of each network, in our case the interaction of the management actors, making them easier to visualize and analyze [17]. Borgatti et al. [18] defines a Simmelian tie as a reciprocally connected pair with mutual ties to third parties and hence, it is an edge embedded in a clique or triple. Simmelian ties help to identify actors with shared interests and common goals by mitigating competition [19]. By providing a conditional triadic interpretation of both network structures helps to get an insight to the social structures of the protected area management networks by highlighting the most redundant and strong ties between the protected area management actors [17].

Romania, a member of the European Union since 2007, designated over 600 Natura 2000 sites, covering more than 22% of its terrestrial surface [20, 21]. However, at national level, a proper implementation of the Habitats and Birds Directives' provisions is still lacking, the main issues being related to deficiency of adaptive conservation planning and management [9, 22].

Given the social focus of Natura 2000 protected areas and large size of sites, the management of Romanian sites is a task managed by many actors including local, regional, and national environmental protection authorities, local and regional administrative authorities, custodians of Natura 2000 sites, local inhabitants, industry, advisory boards, and other local, regional, or national partners (e.g., NGOs) [21]. Romanian protected areas management system (i.e., delegated management to organizations governed by different jurisdictions [23]) might be a good case study to analyze the interaction of protected areas administrators with other organizations involved in management.

To gain insight into how protected areas network relational structure develops around two different types of leading management organizations, our study focuses on two protected sites, namely, Iron Gates Natural Park (administrated by a state-owned enterprise with a public body statute) and Lower Siret Floodplain Natura 2000 (administrated by an NGO) [21]. NGOs are known to have a significant impact on environmental conservation actions and in helping to integrate environmental objectives into policy and practice [24], hence, we hypothesize that the main characteristics of the two analyzed networks differ because they are driven by two different types of egos (public body versus NGO). Furthermore, we hypothesize that the NGO ego-net would be denser than the one driven by a public body, with consequences in the efficiency of any management process.

Obtaining data on the different structural characteristics of each ego-network will help to understand the different patterns of management in terms of cooperation. Furthermore, this paper focuses on the evaluation of protected areas management by considering social patterns of cooperation and highlighting best practices to further contribute to biodiversity conservation [25].

This chapter represents an extension of our previous conference paper [25], in which we briefly described the benefits of characterizing protected areas management using ego-networks. The data-sets used for the ego-network analyses were previously used to describe the current state of the governance network structure [26].

2 Methods

2.1 Study Areas

In 2007, Iron Gates Natural Park became part of Natura 2000 due to its richness of biodiversity of European importance. The park overlaps two Special Protection Areas (SPAs) and one Site of Community Importance (SCIs) [27]. Lower Siret

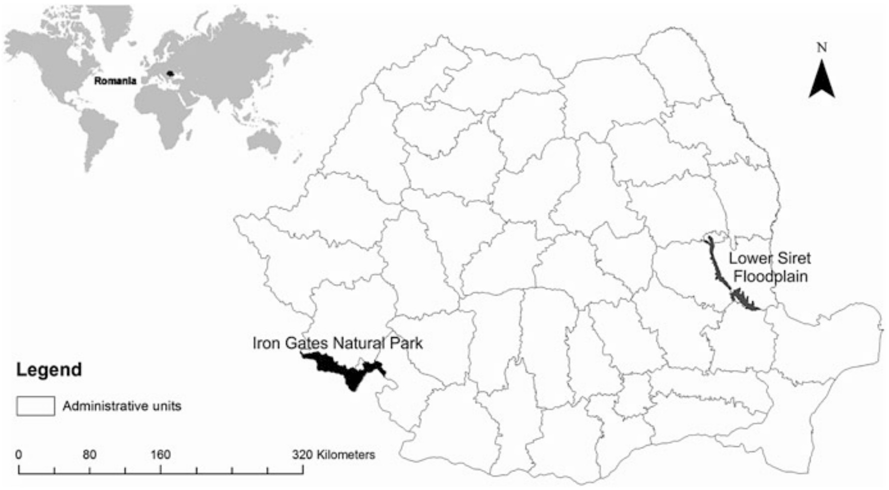


Fig. 1 The location of Iron Gates Natural Park (Southwest) and Lower Siret Floodplain Natura 2000 site (East) within Romania

Floodplain Natura 2000 protects 22 bird species of European interest that are found in Annex I of the Birds Directive, representing a significant hot spot for migratory birds [28].

We selected these two protected areas because they cover the two most common management regimes in Romania, namely, a natural park overlapping Natura 2000 sites, administrated by a public body owned by state (Iron Gates Natural Park, hereinafter, IGNP, administrated by Iron Gates Natural Park Administration) and a Natura 2000 site managed by a regional NGO (Lower Siret Floodplain Natura 2000 site, hereinafter, LSF, administrated by Association for Biodiversity Conservation) (Fig. 1) [25].

2.2 Research Design and Data Collection

We considered as members of the management networks the stakeholders involved in or affected by decisions related with analyzed Natura 2000 sites (*nodes*) and the links between them (*edges* or *links*).

Initially, we identified potential members of the two networks by analyzing the management plans of the two protected areas. Then, between April and June 2016 we surveyed the representatives of 60 organizations from Iron Gates Natural Park and 65 organizations from Lower Siret Floodplain. The survey gathered data on the collaborative relationships established during the management activities. We found that the Iron Gates Natural Park and Lower Siret Floodplain management networks include 99 and 120 organizations, respectively. The difference between the number of actors is mostly because Iron Gates Natural Park is located in two

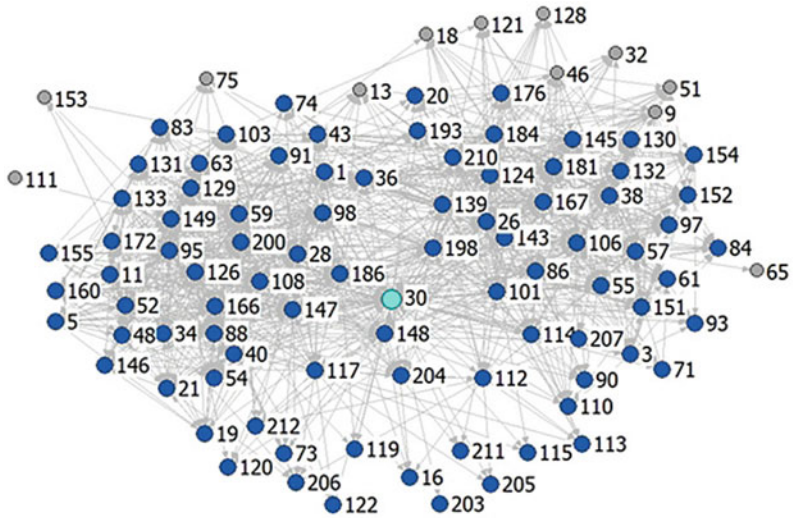


Fig. 2 Case study 1—Iron Gates Natural Park management network (blue circle—ego Iron Gates Natural Park Administration, dark blue circles—alters, grey circles—organizations directly linked with the ego, and grey arrows—connections between actors)

counties (Mehedinti and Caras-Severin), while Lower Siret Floodplain in three counties (Vrancea, Galati, and Braila) [25].

Secondly, we encoded each organization and created, for both sites, adjacency matrices of the directed graphs, with actors and connections between them. We used UCINET 6.620 [18] for all network analyses. Given the fact that the two protected areas are coordinated by distinct types of organizations, we further extracted the ego-networks for the public body Iron Gates Natural Park Administration, hereinafter IGNPA (case study 1—Fig. 2) and for non-governmental organization—Association for Biodiversity Conservation, hereinafter ACDB (case study 2—Fig. 3).

We considered ego-nets as appropriate for our analyses because of their constrained and simpler structure [16]. To reduce the size of the networks, and better identify the potential differences between the two case studies, we further analyzed the structures created around the egos.

Conclusively, our study brings into discussion and analyzes the ego-networks composed of the most important players in the protected areas management.

3 Concepts and Methodology

We analyzed the two case studies by:

1. Comparing the ego-networks structural metrics lead by different types of organization (i.e., NGO versus public body);

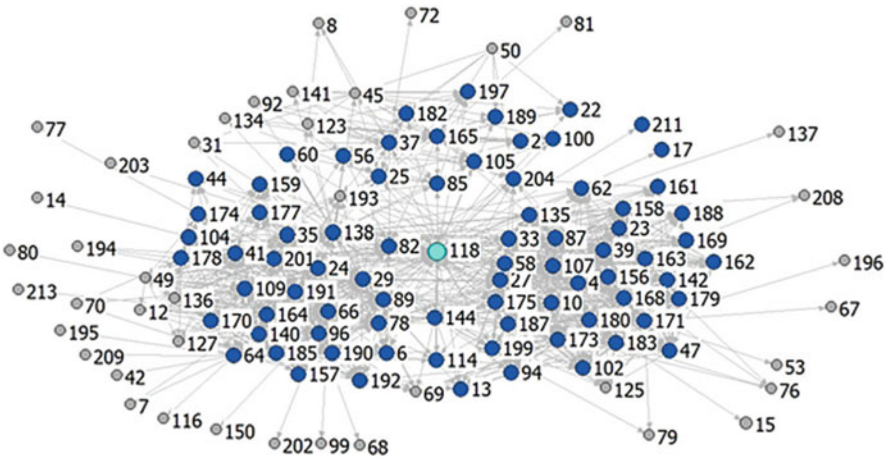


Fig. 3 Case study 2—Lower Siret Floodplain Natura 2000 site management network (blue circle—ego Association for Biodiversity Conservation, dark blue circles—alters, grey circles—organizations not directly linked with the ego, and grey arrows—connections between actors)

2. Correlating the indegree and betweenness values of the networks in three case scenarios:
   (a) before extracting the ego-networks;
   (b) after extracting the ego-networks of the main management organizations;
   (c) after removing the ego from both networks.
3. Analyzing the consequences after removing the ego by applying the Simmelian ties algorithm and Girvan–Newman clustering.

By using UCINET software [18], we calculated for both case studies the main important properties of the ego-network. Hence, we determined the cohesiveness of cooperation among the network management actors. Table 1 summarizes the description of analyzed ego-metrics [34].

The follow-up analysis considers the interpretation of betweenness and indegree values for management actors in the two case studies. Betweenness centrality represents the number of shortest paths that pass through a node, and thus, can be a measure of centrality of an organization, and the indegree represents the number of receiving ties, and can be interpreted as a measure of popularity of an organization [10]. We correlated the indegree and betweenness values and drew conclusions on the properties of each case study, in three distinct forms:

– the complete management network;
– the ego-network of the main administrative management organization;
– the remaining management actors after removing the ego.

Table 1 Interpretation of analyzed ego-networks metrics

Metrics	Interpretation in our analysis
Network size	Number of management actors directly connected with the protected area administration organization [29]
Number of ties	Number of links among all management actors in the ego-network [30]
Network density	Number of connections divided by the number of pairs, namely, the percentage of all possible links in each ego-network [31]
Weak components	A weak component is the largest number of management actors that are connected, disregarding the direction of the link [32]
EgoBetweenness	The percentage of all geodesic paths from neighbor to neighbor that pass through the protected area administration organization [33]
Normalized EgoBetweenness	Compares the actual betweenness of the protected area administration organization to the maximum possible betweenness in neighborhood of the size and connectivity of ego's. The maximum value for betweenness would be achieved where ego is the center of a "star" network, that is, no neighbors communicate directly with one another, and all directed communications between pairs of neighbors go through ego [16]

We carried out this three-step analysis to demonstrate how actors migrate and establish a distinct position within the network when changing the size of the network by removing the egos.

The final step focuses on the consequences of removing the ego by analyzing the role of Simmelian ties in the networks and clustering using Girvan–Newman algorithm, which describes a hierarchical method regularly used to detect communities in complex systems based on edges betweenness [25, 35]. Although slower than other clustering techniques, we choose to use Girvan–Newman algorithm to investigate our ego-networks because it is well documented and often used in popular network analysis programs [36], thus serving well our aim to demonstrate a real-world application of ego-network analysis to differentiate environmental management structures. Giving the potential to transfer the results obtained for an ego-net to a complete network [29], we further extracted substructures for each case study consisting of ties that are strong and redundant [17]. To learn what would happen with the management networks (thus, the whole management of the protected area) if the egos (i.e., protected areas administration) would disappear, we applied the Simmelian ties algorithm, using the triples method "mutual friends" [37] between each pair of actors, before and after removing the protected area management administration main actor (ego of each case study). Social ties embedded within triads are more stable over time, stronger, and more durable [35]. In our case, analyzing this is important, because informational flow can be obstructed if involved management organizations are not being proactive or are acting opportunistically and avoid exchanging knowledge and information with other actors of the network [19].

We illustrate the results of this application using NetDraw [18] and VosViewer [23]. Additionally, to identify community structure in both networks, we grouped the actors in clusters using Girvan–Newman algorithm [38]. Typically, this hierarchical clustering is used to remove the highest betweenness edges until the network falls apart; however, our principal objective is to compare the two managements of the protected areas by removing the ego. In our paper, we reported the results of the Girvan–Newman algorithm by grouping the management actors in clusters (i.e., corresponding to maximum three counties overlapping the protected areas), without specifying how many groups shall be formed or assigning restrictions on their size [38].

4 Research Findings and Discussion

4.1 Structure of the Ego-Networks

After the extraction of the ego from both analyzed management networks (i.e., organizations in charge with administration), we observed that the size of Lower Siret Floodplain management ego-network (case study 2) registered a significant reduction, from 120 to 78, because 35% of the network actors were not directly connected with the ego (ACDB). In terms of ego-network size (Table 2), Iron Gates Natural Park management network (case study 1) retains more than 87% of the actors of the initial network including a larger number of ties, reduction from 99 to 87 actors.

Our results indicate that the ego-network of the Iron Gates Natural Park is denser with lesser number of weak components. Conversely, Lower Siret Floodplain ego-network has a higher betweenness, whereas the weighted overall graph clustering coefficient appears to be similar for both analyzed cases (see Table 2).

Even though the number of organizations participating in the management of Lower Siret Floodplain is higher than in the Iron Gates Natural Park, the ego is less connected with the other management actors from the territory. This is expected

Table 2 Properties of the ego-network case studies

Metrics	Case study 1: Iron Gates Natural Park	Case study 2: Lower Siret Floodplain
Initial size of the management network	99	120
Size of management EgoNet (including ego)	87	78
Directed ties	1236	844
Network density	16.91	14.42
Weak components	2	3
Number of weak components divided by sizes	2.33	3.90
EgoBetweenness	1571.07	1903.81
Normalized EgoBetweenness	21.39	32.53

since most organizations involved in or affected by decisions related to management of analyzed protected areas are public bodies, and usually organizations with different jurisdictions (i.e., public bodies vs. NGOs have a lower tendency to cooperate, mostly due to the difference between governing norms) [39, 40].

4.2 *Correlation Between the Indegree and Betweenness Values of the Networks*

By plotting indegree versus betweenness for the two case studies, we observed a dissimilar pattern of movement of the most important actors in terms of betweenness and indegree while changing the network magnitude (Figs. 4 and 5). Thus, for Iron Gates Natural Park (Fig. 4, we detected several actors that have redundant connections and significant impact within the communication flow (e.g., EPA MH—Environmental Protection Agency Mehedinti county and EPA CS—Environmental Protection Agency Caras-Severin county)), while for the Lower Siret Floodplain (Fig. 5) there are actors (e.g., Adjud Town Hall) that appear to establish a crucial position when analyzing the hole network but lose importance when extracting the ego-network.

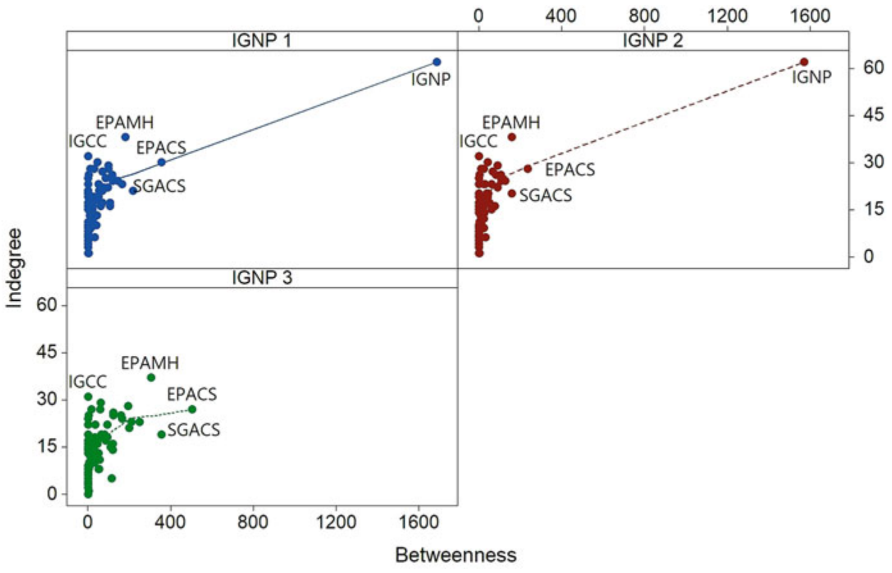


Fig. 4 Indegree versus betweenness of organizations active in Iron Gates management network: IGNP 1—before extracting the ego-network; IGNP 2—after extracting the ego-network; IGNP 3—after removing the ego [25]

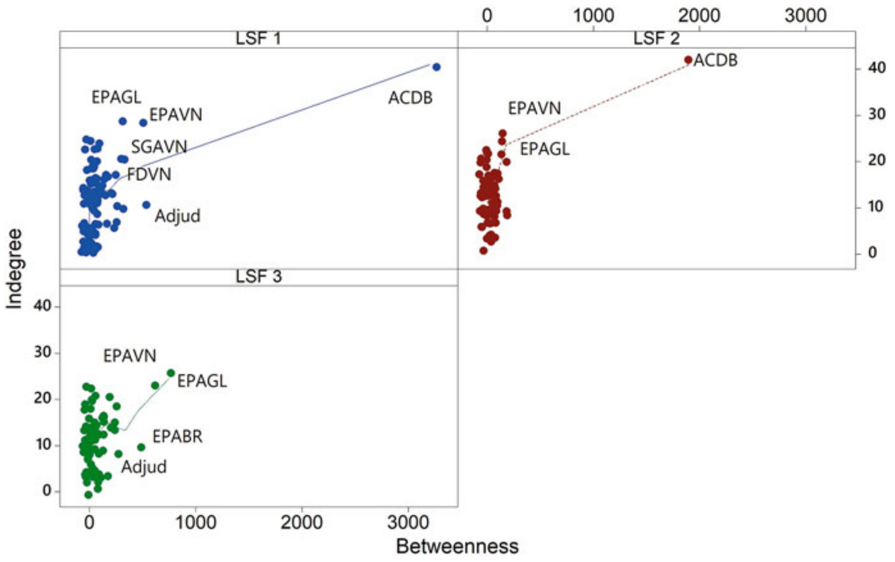


Fig. 5 Indegree versus betweenness of organizations active in Lower Siret management network: LSF 1—before extracting the ego-network; LSF 2—after extracting the ego-network; LSF 3—after removing the ego

This confirms the potential weakness of Lower Siret Floodplain management network, where actors are connected only with protected area administration and do not communicate directly. In the case of Iron Gates Natural Park network, the communication between actors can be efficient even if the protected area administrator is not responsive. This can be interpreted as an ability of other public bodies from the respective management network to act as redundant connections for a public body protected area administration, while the NGOs administrating a protected area must communicate actor by actor to perform management activities in their area of interest, particularly in environmental protection and conservation. This analysis confirms the pattern resulted from interpreting the ego-networks properties, and most importantly, underlines that the NGOs administrating a protected area must strengthen the cooperation with key public body actors, such as environmental protection agencies and local and regional administrative authorities.

4.3 Consequences of Removing the Ego: Role of Simmelian Ties and Girvan–Newman Clustering

Figures 6 and 7 illustrate the results after the integration of the Simmelian ties analysis and filtering the clusters using Girvan–Newman algorithm for both case studies and with and without egos. By analyzing case study 2 (Lower Siret

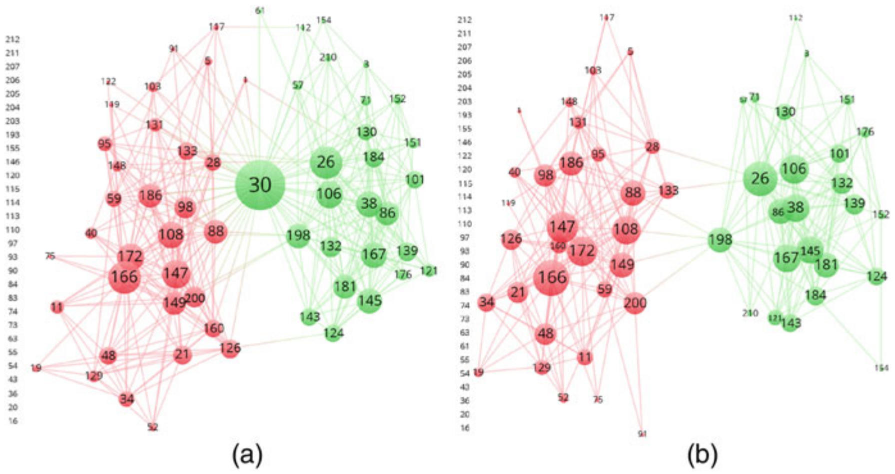


Fig. 6 Simmelian ties and Girvan–Newman clustering of Iron Gates Natural Park management network (**a**) before and (**b**) after removing ego. Size of the nodes is given by the total links strength

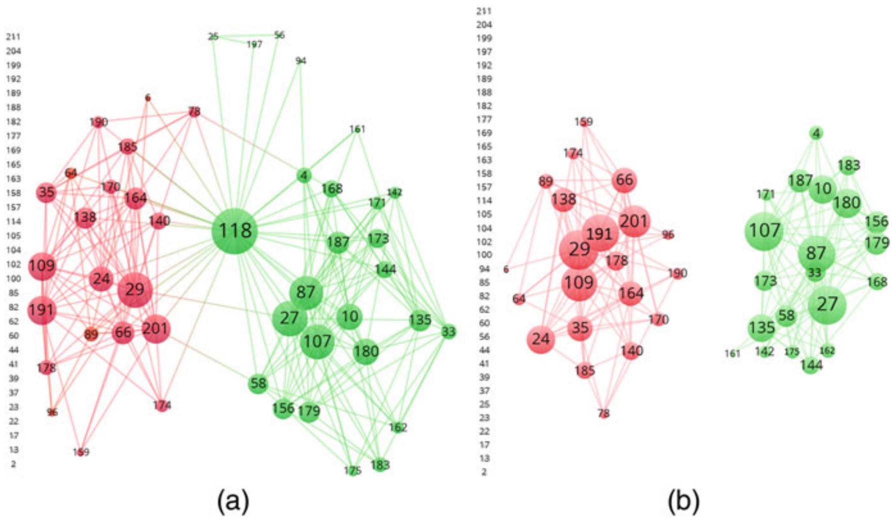


Fig. 7 Simmelian ties and Girvan Newman clustering of Lower Siret Floodplain management network (**a**) before and (**b**) after removing ego. Size of the nodes is given by the total links strength [25]

Floodplain), we found out that our hypotheses are contradicted by the results. Unexpectedly, Lower Siret Floodplain network, which has an NGO as an ego, has a greater potential for falling apart (Fig. 7a, b).

After removing the ego, the Lower Siret Floodplain network falls apart, because the management organizations, except the ego, have roles only at local level and do not ensure communication within the entire network, failing to ensure a flow in the management processes. We were expecting that protected areas administrated by an NGO to be better connected to other organizations than one governed by a public body [5, 41]. From this perspective, the Iron Gates Natural Park network has a more effective collaborative management, adopting the premise that the managerial organizations within a protected area should not become too dominant but recognize others as collaborative partners [42]. This, along with enhanced communication flow, sharing responsibilities and transfer of knowledge, may be the key for successful management of protected areas.

As it can be seen in Figs. 6 and 7, by taking Simmelian ties into account after clustering the organizations, we defined three categories of network actors, this way predicting the future of a management network if ego (main management organization) would disappear. Figure 7 reinforces our prior results (structure of the ego-networks and correlation between the indegree and betweenness values of the networks) by identifying in Iron Gates Natural Park management network, stronger Simmelian bridges, the most important management actors remaining connected and thus ensuring a faster information flow and more qualitative associated management processes.

5 Conclusions and Future Work

Integrating social network and ego-network analyses within environmental management provides an overview of the actual patterns of the structure, but the results obtained for each environmental management network could not be generalized to all protected areas, the findings being beneficial for stakeholders and practitioners in the field.

The current study contributes to our understanding of the linkage between protected areas management actors. By analyzing ego-networks structural metrics, correlating the indegree and betweenness values, and applying the Simmelian ties algorithm and Girvan–Newman clustering after removing the ego, we successfully achieved a diagnosis of main social structures in the management of two Natura 2000 sites, representative for the management of protected areas in Romania, namely, Iron Gates and Lower Siret Floodplain. Using ego-network analysis, we have succeeded in obtaining significant information and contribute to the environmental management field by showing a way to facilitate more accurate and efficient management analysis [43].

Our results demonstrate large variations in ego-networks metrics, indicating that the ego-networks properties depend on the ability of the ego to connect to organization with different jurisdictions [44]. In our case, the network coordinated by an NGO (Lower Siret Floodplain) is more fragmented, and such, the collaboration and information flows are strongly influenced by the existence of one key actor only. Such networks are more vulnerable to disruptions of collaborations [45], for example, when the management body tries to enforce more strict regulations on resource management to protect the biodiversity. Without taking in consideration the management performance, after comparing the two management networks, we could consider that Iron Gates Natural Park management network is an example of best practice in structural terms.

Examining the effect of tie strength presents an opportunity to predict what role the stakeholders play within the management of the protected areas. Additionally, using embedded ties and link strengths showed for Lower Siret Floodplain a higher potential for collapse if removing the ego from the network.

Our results about network formation and structure could be further developed by comparing network structure over time [46] and be extended to other protected area management, presenting informative potential for managers. Considering expanding our results to other protected areas, the level of cooperation and actions will positively change so that the conservation targets may be achieved without stimulating social conflicts [47, 48]. In line with previous findings [15], using social network analysis in protected area management can help involved actors to understand the potential risks of weak network components and predict arrangements that can undermine conservation efforts.

Interpreting Simmelian ties findings could play a defining role in training and forming future leaders. From this point of view, this technique could have a significant potential to establish the development of future leaders within environmental management networks.

We demonstrated that network analysis can contribute to improve protected areas management and may be a useful tool for systematic conservation planning [49]. Our findings could be further used also to minimize the protected area network's vulnerabilities and predict the potential for large-scale failure. From this point of view, using ego-network analysis within the protected areas management can be a starting point to adapt over time and recover after a network disruption and hence contribute to the implementation of Habitats and Birds Directive. This is why, network analysis may play a defining role in good resources management promotion [50, 51]. Nevertheless, further work is needed to create an accessible streamlined methodology so that the protected areas managers use the insights that social network analysis could provide to conservation planning.

Acknowledgements We would like to thank Iron Gates Natural Park and Lower Siret Floodplain administrations and Iulia Viorica Miu for the data collection process, and to Edward F. Rozylowicz for proofreading the manuscript. This work was supported by a grant of the Romanian National Authority for Scientific Research, CNCS—UEFISCDI, PN-III-P4-ID-PCE-2016-0483 (http//www.uefiscdi.ro).

References

1. European Commission: Council Directive 92/43/EEC of 21 May 1992 on the conservation of natural habitats and of wild fauna and flora (Habitats Directive). Off. J. Eur. Union L. **206**, 7–50 (1992)
2. European Commission Directive 2009/147/EC of the European Parliament and of the Council of 30 November 2009 on the conservation of wild birds: Off. J. Eur. Union L. **20**, 7–25 (2009)

3. European Commission: Environmental impact assessment of projects: rulings of the court of justice. Report, European Commission (2013)
4. Hanspach, J., Hartel, T., Milcu, A.I., Mikulcak, F., Dorresteijn, I., Loos, J., von Wehrden, H., Kuemmerle, T., Abson, D., Kovacs-Hostyanszki, A., Baldi, A., Fischer, J.: A holistic approach to studying social-ecological systems and its application to southern Transylvania. Ecol. Soc. **19**(4), art32 (2014). https://doi.org/10.5751/Es-06915-190432
5. Nita, A., Rozylowicz, L., Manolache, S., Ciocanea, C.M., Miu, I.V., Popescu, V.D.: Collaboration networks in applied conservation projects across Europe. PLoS One **11**(10), e0164503 (2016). https://doi.org/10.1371/journal.pone.0164503
6. Blicharska, M., Orlikowska, E.H., Roberge, J.M., Grodzinska-Jurczak, M.: Contribution of social science to large scale biodiversity conservation: a review of research about the Natura 2000 network. Biol. Conserv. **199**, 110–122 (2016). https://doi.org/10.1016/j.biocon.2016.05.007
7. Popescu, V.D., Rozylowicz, L., Niculae, I.M., Cucu, A.L., Hartel, T.: Species, habitats, society: an evaluation of research supporting EU's Natura 2000 network. PLoS One **9**(11), e113648 (2014). https://doi.org/10.1371/journal.pone.0113648
8. Ioja, I.C., Hossu, C.A., Nita, M.R., Onose, D.A., Badiu, D.L., Manolache, S.: Indicators for environmental conflict monitoring in Natura 2000 sites. Procedia Environ. Sci. **32**, 4–11 (2016). https://doi.org/10.1016/j.proenv.2016.03.007
9. Pellegrino, D., Schirpke, U., Marino, D.: How to support the effective management of Natura 2000 sites? J. Environ. Plan. Manag. **60**(3), 383–398 (2017). https://doi.org/10.1080/09640568.2016.1159183
10. Borgatti, S.P., Everett, M.G., Johnson, J.C.: Analyzing Social Networks, 2nd edn. SAGE, London (2017)
11. Bodin, O., Robins, G., McAllister, R.R.J., Guerrero, A.M., Crona, B., Tengo, M., Lubell, M.: Theorizing benefits and constraints in collaborative environmental governance: a transdisciplinary social-ecological network approach for empirical investigations. Ecol. Soc. **21**(1), art40 (2016). https://doi.org/10.5751/Es-08368-210140
12. Scarlett, L., McKinney, M.: Connecting people and places: the emerging role of network governance in large landscape conservation. Front. Ecol. Environ. **14**(3), 116–125 (2016). https://doi.org/10.1002/fee.1247
13. Rey, V., Groza, O., Patroescu, M., Ianos, I.: Atlas de la Roumanie: dynamiques du territoire. Reclus. La Documentation française (2007)
14. Kitsak, M., Gallos, L.K., Havlin, S., Liljeros, F., Muchnik, L., Stanley, H.E., Makse, H.A.: Identification of influential spreaders in complex networks. Nat. Phys. **6**(11), 888–893 (2010). https://doi.org/10.1038/Nphys1746
15. Alexander, S.M., Andrachuk, M., Armitage, D.: Navigating governance networks for community-based conservation. Front. Ecol. Environ. **14**(3), 155–164 (2016). https://doi.org/10.1002/fee.1251
16. Everett, M., Borgatti, S.P.: Ego network betweenness. Soc. Netw. **27**(1), 31–38 (2005). https://doi.org/10.1016/j.socnet.2004.11.007
17. Nick, B., Lee, C., Cunningham, P., Brandes, U.: Simmelian backbones: amplifying hidden homophily in Facebook networks. In: 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM), pp. 531–538 (2013). https://doi.org/10.1145/2492517.2492569
18. Borgatti, S.P., Everett, M.G., Freeman, L.C.: Ucinet for Windows: Software for Social Network Analysis. Analytic Technologies, Harvard (2002)
19. Tortoriello, M., Krackhardt, D.: Activating cross-boundary knowledge: the role of Simmelian ties in the generation of innovations. Acad. Manage. J. **53**(1), 167–181 (2010). https://doi.org/10.5465/Amj.2010.48037420
20. Manolache, S., Ciocanea, C.M., Rozylowicz, L., Nita, A.: Natura 2000 in Romania - a decade of governance challenges. Eur. J. Geogr. **8**, 24–34 (2016)

21. Ioja, C., Patroescu, M., Rozylowicz, L., Popescu, V., Verghelet, M., Zotta, M.L., Felciuc, M.: The efficacy of Romania's protected areas network in conserving biodiversity. Biol. Conserv. **143**(11), 2468–2476 (2010). https://doi.org/10.1016/j.biocon.2010.06.013
22. Nita, M.R., Niculae, I.M., Vanau, G.O.: Integrating spatial planning of protected areas and transportation infrastructures. In: Using Decision Support Systems for Transportation Planning Efficiency, pp. 311–329. Engineering Science Reference, Hershey (2016)
23. van Eck, N.J., Waltman, L.: Software survey: VOSviewer, a computer program for bibliometric mapping. Scientometrics **84**(2), 523–538 (2010). https://doi.org/10.1007/s11192-009-0146-3
24. Runhaar, H.: Tools for integrating environmental objectives into policy and practice: what works where? Environ. Impact Assess. Rev. **59**, 1–9 (2016). https://doi.org/10.1016/j.eiar.2016.03.003
25. Nita, A., Manolache, S., Ciocanea, C., Rozylowicz, L.: Characterizing protected areas management using ego-networks. In: 2017 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining, pp. 642–643 (2017). https://doi.org/10.1145/3110025.3110079
26. Manolache, S., Nita, A., Ciocanea, C.M., Popescu, V.D., Rozylowicz, L.: Power, influence and structure in Natura 2000 governance networks. A comparative analysis of two protected areas in Romania. J. Environ. Manage. **212**, 54–64 (2018). https://doi.org/10.1016/j.jenvman.2018.01.076
27. Iron Gates Natural Park Administration: Planul de management al Parcului Natural Porţile de Fier. Report, RNP Romsilva-Administratia Parcului Natural Portile de Fier (2013)
28. Association for Biodiversity Conservation: Planul de management al ROSPA0071 Lunca Siretului Inferior si al ariilor naturale protejate suprapuse. Report, ACDB (2015)
29. Chang, A.X.: Analysis of Email Ego Networks. An exploratory study of ego networks in an email network. Snap.Stanford.Edu (2010)
30. Harrigan, N., Achananuparp, P., Lim, E.P.: Influentials, novelty, and social contagion. The viral power of average friends, close communities, and old news. Soc. Netw. **34**(4), 470–480 (2012). https://doi.org/10.1016/j.socnet.2012.02.005
31. Bellotti, E.: Getting funded. Multi-level network of physicists in Italy. Soc. Netw. **34**(2), 215–229 (2012). https://doi.org/10.1016/j.socnet.2011.12.002
32. Hertzberg, V.S., Baumgardner, J., Mehta, C.C., Elon, L.K., Cotsonis, G., Lowery-North, D.W.: Contact networks in the emergency department: effects of time, environment, patient characteristics, and staff role. Soc. Netw. **48**, 181–191 (2017). https://doi.org/10.1016/j.socnet.2016.08.005
33. Epasto, A., Lattanzi, S., Mirrokni, V., Sebe, I.O., Taei, A., Verma, S.: Ego-net community mining applied to friend suggestion. Proc. VLDB Endowment **9**(4), 324–335 (2015). https://doi.org/10.14778/2856318.2856327
34. Hanneman, R., Riddle, M.: Introduction to Social Network Methods. University of California, Riverside (2005)
35. Girvan, M., Newman, M.E.: Community structure in social and biological networks. PNAS **99**(12), 7821–7826 (2002). https://doi.org/10.1073/pnas.122653799
36. Newman, M.E.J.: Modularity and community structure in networks. Proc. Natl. Acad. Sci. **103**(23), 8577–82 (2006). https://doi.org/10.1073/pnas.0601602103
37. Wellington, J.F., Lewis, S.A.: A method for evaluating the funding of components of natural resource and conservation projects. Environ. Impact Assess. Rev. **57**, 40–45 (2016). https://doi.org/10.1016/j.eiar.2015.10.009
38. Newman, M.E.J.: Detecting community structure in networks. Eur. Phys. J. B Condensed Matter Complex Syst. **38**(2), 321–330 (2004)
39. Ostrom, E.: Understanding Institutional Diversity. Princeton University Press, New Jersey (2005)
40. Ostrom, E.: Governing the Commons: The Evolution of Institutions for Collective Action. Cambridge University Press, Cambridge (1990)

41. Rozylowicz, L., Nita, A., Manolache, S., Ciocanea, C.M., Popescu, V.D.: Recipe for success: a network perspective of partnership in nature conservation. J. Nat. Conserv. **38**, 21–29 (2017). https://doi.org/10.1016/jnc.2017.05.005
42. Fliervoet, J.M., Geerling, G.W., Mostert, E., Smits, A.J.: Analyzing collaborative governance through social network analysis: a case study of river management along the Waal river in The Netherlands. Environ. Manag. **57**(2), 355–67 (2016). https://doi.org/10.1007/s00267-015-0606-x
43. Li, C., Lin, S.: Egocentric information abstraction for heterogeneous social networks. In: 2009 International Conference on Advances in Social Network Analysis and Mining, pp. 255–260 (2009). https://doi.org/10.1109/ASONAM.2009.38
44. Ionita, A., Stanciu, E.: Protected area governance in Eastern Europe. Report, BfN-Skripten 360 (2015)
45. Horning, D., Bauer, B.O., Cohen, S.J.: Missing bridges: Social network (dis)connectivity in water governance. Util. Policy **43**, 59–70 (2016). https://doi.org/10.1016/j.jup.2016.06.006
46. Karsai, M., Perra, N., Vespignani, A.: Time varying networks and the weakness of strong ties. Sci. Rep. **4**, 4001 (2014). https://doi.org/10.1038/srep04001
47. Hossu, C.A., Ioja, I.C., Nita, M.R., Hartel, T., Badiu, D.L., Hersperger, A.M.: Need for a cross-sector approach in protected area management. Land Use Policy **69**, 586–597 (2017). https://doi.org/10.1016/j.landusepol.2017.10.012
48. Hossu, C.A., Ioja, I.C., Susskind, L.E., Badiu, D.L., Hersperger, A.M.: Factors driving collaboration in natural resource conflict management: evidence from Romania. Ambio **47**, 816–830 (2018). https://doi.org/10.1007/s13280-018-1016-0
49. Sayles, J.S., Baggio, J.A.: Who collaborates and why: assessment and diagnostic of governance network integration for salmon restoration in Puget Sound, USA. J. Environ. Manage. **186**(1), 64–78 (2017). https://doi.org/10.1016/j.jenvman.2016.09.085
50. Keskitalo, E.C., Baird, J., Laszlo Ambjornsson, E., Plummer, R.: Social network analysis of multi-level linkages: a Swedish case study on Northern Forest-Based sectors. Ambio **43**(6), 745–58 (2014). https://doi.org/10.1007/s13280-014-0492-0
51. Berardo, R., Heikkila, T., Gerlak, A.K.: Interorganizational engagement in collaborative environmental management: evidence from the South Florida ecosystem restoration task force. J. Public Adm. Res. Theory **24**(3), 697–719 (2014). https://doi.org/10.1093/jopart/muu003

An Evolutionary Approach for Detecting Communities in Social Networks

Koray Ozturk, Faruk Polat, and Tansel Özyer

Abstract Recent advancements and increasing use of social networking applications have made extensive amounts of data available. Because of this, exploring new and effective methods for mining and analyzing social network data is needed. In our work, a method inspired by evolutionary approach is proposed to find communities in social networks. A genetic algorithm, which is able to detect communities without needing the number of communities at the beginning of the algorithm, has been formulated and compared with other community detection methods to prove its accuracy, efficiency, and effectiveness. In addition, experiments using Newman's spectral clustering method as a preprocessing step to our modified genetic algorithm have been done and seen producing better results for large datasets.

Keywords Social networks · Community detection · Genetic algorithms

1 Introduction

The area of social network has gained interest of researchers after websites such as Facebook, LinkedIn, Twitter, and Google+ came out. Now, these social networking websites have become an important part of our daily life, they are tools for not only to know what our friends are doing but also to market products, organize events, and spread thoughts. Social networks have two main properties called *entity* and *relationship* forming the data. Entities might be "people" and the relationships might be the "friendship" of these people like in Facebook and like most of the other

K. Ozturk (✉) · F. Polat
Middle East Technical University, Ankara, Turkey
e-mail: koray.ozturk@ceng.metu.edu.tr; polat@ceng.metu.edu.tr

T. Özyer
Department of Computer Engineering, TOBB University of Economics and Technology, Ankara, Turkey
e-mail: ozyer@etu.edu.tr

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_2

social websites, but they are not limited to "people" and "friendship." For example, organizations, websites, products could be counted as entities, while business, trade, collaboration activities are among relationships. Relationships can be all-or-nothing as in Facebook that if you are friend with someone or not, or can be a degree of friendship as in Google+. Even though social networks and their analysis have been a widespread research area in sociology [1, 2] for decades, late advancements in Internet and computer applications have made extensive amount of real world data accessible to analyze and process for researchers. Since real world social networks might be very large in size, even exceeding billion of nodes, there is a need for changing how to handle analyzing and processing networks. Therefore, a large number of new methods are being produced to find efficient and effective methods for analysis of real world social networks [3–8].

In a randomly created network, edges are mostly distributed equally and because of this, the degrees of nodes are expected to be very close [9]. On the other hand, degree distributions of real world networks are not homogeneous, within a group of entities more edges might be placed while within some other group of entities there might be less edges [10]. This aspect of social networks whose edges within some specific group of entities are denser is called community structure [11]. As may be expected, the entities of a social network are placed into communities in which the relationships between the entities are dense, on the other hand, the relationships between the entities of different communities are scarce. Community detection is significant, because a community might be a small version of whole graph, which shows the very similar characteristics of it. Therefore, examining a few communities might enable us to understand the whole network. This feature is very useful especially when network is a very large real world data. Community detection is beneficial not only for detecting communities of individuals but also for advancements in commercial and academic areas. For example, dividing citation network into communities can help researchers who are looking for cooperation for a specialized field [12, 13]. In the studies related to social networks, the topic of community detection has been discussed largely in the context of block models which are the divisions of the networks into the basic blocks according to some criteria. If we have the block model, we can have communities [14, 15]. Our study focuses on discovering communities in social networks, and we propose a method for detecting communities in social networks making use of the evolutionary approach.

Graph is an appropriate tool for modeling the discrete structures such as social networks. If we define a graph as $G = (V, E)$, then V is the set of vertices, which can be called as *nodes* in this work, and E is the set of edges that if an edge exists between the vertices V_i and V_j then we can say that vertices V_i and V_j are related to each other and the edge between them is shown as E_{ij}. While modeling social networks as graphs, an entity is modeled as a vertex and an edge connecting two nodes indicates the relationship between the nodes. Undirected graphs are the best and the most natural exhibition of the social networks, so we will be using undirected graphs in this work to model social networks. In terms of undirected graphs, E_{ij} is the same as E_{ji}. Furthermore, in our work, graphs do not include loops and are non-reflexive, meaning that vertices are not related to themselves. Also, multiple edges from one node to another do not exist. We can represent a graph either visually or with an adjacency matrix A, a VxV square matrix, where vertices are in rows and columns, and numbers in the matrix indicate the existence of edges such as if E_{ij} exists, then the value of entry a_{ij} is 1, else 0. For unweighted graphs, all entries are 0 or 1; for weighted graphs, the adjacency matrix contains the values of the weights. Since our graph is non-reflexive, diagonal of the adjacency matrix A contains only zeros. In Fig. 9, an undirected sample graph with communities is shown.

2 Related Work

Plenty of algorithms exist partitioning networks into parts that we call these parts as communities. Most of them work well on artificially produced datasets, or on real world datasets of which communities are known. However, evaluating the quality of the community structures is a significant issue in case of working on real world datasets and not knowing the communities previously. It is generally agreeable that edges between individuals are denser in a well-defined community structure. A node must have most of its connections with the nodes which are in the same community and must have none or very few connections with the nodes which belong to other communities.

In the work of Radicchi et al. [16], a quantitative measure for evaluation of communities is proposed. However, it is also stated that quantitative measures are subjective and cannot be exactly accurate for now. Lately, *Modularity* concept proposed by Newman and Girvan [17] is accepted as a qualification measure for communities. This measure is based on the previous work of Newman which focuses on assortative mixing [18]. *Modularity* calculation is shown in the following equation:

$$Q = \sum_{i} \left(e_{ii} - a_i^2 \right) \tag{1}$$

where i is the number of communities, e_{ii} is the fraction of edges to the total number of edges in the network that have both sides inside the community, and a_i is the fraction of total number of edges that have at least one side inside the community to the total number of edges in the network. When we are counting edges for a_i, if one side of the edge is inside the community i and the other side is in another community, then we count that edge as 0.5. If both sides of the edge were in the community i, then we would count it as 1. After the calculation, modularity score Q takes a value between -1 and 1. Q values which are closer to 1 have better community structures.

Girvan and Newman algorithm (GNA) [11] introduces the use of modularity score to measure the efficiency and effectiveness of the algorithm. In GNA, the edges that are most between the communities are focused and removal of the most between edges progressively from the original graph is aimed. To achieve this, *betweenness* of each edge in the network must be found by first visiting each node using breadth first search and then calculating shortest paths from each node to other nodes through edges. Betweenness of an edge E_{ij} is the number of the shortest paths between the nodes x_i and x_j of the graph $G = (V, E)$ that pass through this edge where $x_i, x_j \in V, 1 \leq i \neq j \leq |V|$. If there is more than one shortest path between x_i and x_j, then their fraction is taken. We can outline the steps of the GNA as follows:

1. Find the betweenness scores of all edges in the network.
2. Choose the edge with the highest betweenness and remove it from the network.
3. Recalculate betweenness for all remaining edges.
4. Repeat from step 2.

Slowness of GNA, which has a running time of O(n^3) in sparse graphs, is an important difficulty, such that it is considered inefficient for networks which have more than 10,000 nodes. Although it is inconvenient for large networks, several successful versions of Girvan–Newman algorithm are used in different publications by tailoring it for the datasets used. Holme et al. [19] used GNA for metabolic networks and Gleiser and Danon [20] used itto split early jazz musicians into communities that musicians who collaborated are aimed to be in the same community. Guimera et al. [21] also used GNA on a dataset consisting of the e-mail network of a university.

The algorithm of Duch and Arenas (DA) [22], accepting concept of modularity proposed by Newman and Girvan [11] as the measurement method for defining community structures, presents a novel method which tries to maximize the modularity score Q of the large complex networks to detect community structures. In the initial step, all nodes in the graph are placed into two communities randomly such that both of the communities contain same or one different number of nodes. In every step, the node having lower fitness value, which is the modularity score of a node in its community, is moved from one community to the other and then fitness of the nodes' neighbor nodes is recalculated. This process is repeated until a maximized modularity score is achieved. After two separate communities are formed, all of the edges between them are removed and the same process repeated on each community until modularity score cannot be improved further.

The algorithm proposed by Clauset et al. (CA) [23] is a kind of agglomerative hierarchical clustering method which also uses modularity as a measure for optimization of community structures. In the initial phase of the algorithm, n communities are created, where n is the number of the nodes in the network. All of the nodes are distributed randomly to communities which do not contain another node. In other words, each community contains sole node at the beginning. Communities are combined in pairs in each step to form one unified community, picking out the combination that results in the greatest increase or least decrease in modularity score. Merging process repeats until a tree that shows the order of joins called a dendrogram is created. At the end, a cut through this dendrogram reveals the community structures of the network. Depending on the level we cut, we might get small or large communities. Although an exact definition was not made for the place of the cut in CA, putting the cut where it will maximize the modularity score Q of the network is recommended to get better outcomes.

Graph partitioning is a common technique that divides the graph into groups which have similar size, while trying to minimize the number of the edges between these groups. Most of the graph partitioning methods are based on dividing the graphs into two separate groups iteratively: The *spectral bisection* method [24, 25] which uses Laplace matrix of the graph and eigenvectors of it and *Kernighan Lin algorithm* [26] which aims to optimize community structure over an initial partition of the graph in a greedy way. Although the main characteristic of the spectral bisection method is partitioning the graph into two subgraphs, this is a disadvantage when more than two communities exist. If we need to find more than two communities, spectral bisection must be repeated iteratively on the subgraphs that this approach does not guarantee the fulfillment of expected results. Also, deciding where to stop dividing the graphs is important. The spectral bisection method runs in O(n^3) time. The Kernighan Lin algorithm which is a specialized approach to spectral bisection is a greedy optimization algorithm, and it tries to maximize a benefit function. The benefit function is the sum of edges within groups minus the sum of the edges between groups. The Kernighan Lin algorithm is as follows:

1. Start with the initial partition of the graph into two groups. Size of the groups must be predefined. Nodes might be assigned to the groups randomly.
2. Consider all possible pair of nodes where one node is chosen from each group and calculate the change in the benefit function in case if we swap them.
3. The swap that maximizes the benefit function is chosen and swap is done. Step 2 and 3 are repeated until all nodes in one of the groups have been swapped once.
4. The sequence of swaps that were made is reexamined and the point during this sequence at which Q was highest is found. This is taken to be the bisection of the graph.

The necessity of providing the sizes of the communities at the initialization phase is the main disadvantage of the Kernighan Lin algorithm such that outcomes are highly depended on the size and configurations given at the beginning, making the Kernighan Lin algorithm unsuitable for real world datasets. In addition, it suffers the same disadvantage with other spectral bisection methods: Network is always divided into two communities, and although division into more than two communities can be done iteratively, we don’t know where to stop the iteration for the best division. After some advancements done by Blondel et al. [27] and Mucha et al. [28], Kernighan Lin algorithm is extended that the need of predetermining the number and size of communities at the beginning is overcome by moving a single node to other communities at a time. However, this advancement has shortcomings causing the algorithm to consume more time and perform poor at detecting communities.

Newman’s spectral algorithm [29] is another method for graph partitioning, which is referred to as NSA in this work. NSA uses some important tools from matrix theory, which can be named as spectral methods, to formulate the problem

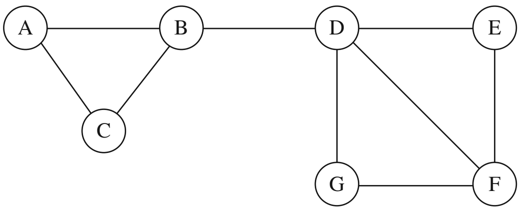


Fig. 1 Example of a small social network

of partitioning a graph to minimize the number of edges that connect different components. Given a graph, we would like to divide the nodes into two sets so that the cut, or set of edges that connect nodes in different sets, is minimized.

Newman's spectral algorithm uses the adjacency matrix to divide a graph into two good partitions that the adjacency matrix has a 1 in row i and column j if there is an edge between nodes i and j, and 0 otherwise. Adjacency matrix A of a small social network example which is presented in Fig. 1 is as follows:

$$A = \begin{bmatrix} 0&1&1&0&0&0&0 \\ 1&0&1&1&0&0&0 \\ 1&1&0&0&0&0&0 \\ 0&1&0&0&1&1&1 \\ 0&0&0&1&0&1&0 \\ 0&0&0&1&1&0&1 \\ 0&0&0&1&0&1&0 \end{bmatrix}$$

The second matrix we need is the degree matrix for a graph. This graph has entries only on the diagonal. The entry for row and column i is the degree of the ith node. The degree matrix for the graph of Fig. 1 is shown below. For instance, the entry in row 4 and column 4 is 4 because node D has edges to four other nodes. The entry in row 4 and column 5 is 0, because that entry is not on the diagonal.

$$D = \begin{bmatrix} 2&0&0&0&0&0&0 \\ 0&3&0&0&0&0&0 \\ 0&0&2&0&0&0&0 \\ 0&0&0&4&0&0&0 \\ 0&0&0&0&2&0&0 \\ 0&0&0&0&0&3&0 \\ 0&0&0&0&0&0&2 \end{bmatrix}$$

Suppose our graph has adjacency matrix A and degree matrix D. Our third matrix, called the *Laplacian matrix*, is $L = D - A$, the difference between the degree matrix and the adjacency matrix. That is, the Laplacian matrix L has the same entries as D on the diagonal. Off the diagonal, at row i and column j, L has -1 if there is an edge between nodes i and j and 0 if not. The Laplacian matrix for the graph of

Fig. 1 is shown below. Notice that each row and each column sums to zero, as must be the case for any Laplacian matrix.

$$L = \begin{bmatrix} 2 & -1 & -1 & 0 & 0 & 0 & 0 \\ -1 & 3 & -1 & -1 & 0 & 0 & 0 \\ -1 & -1 & 2 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 4 & -1 & -1 & -1 \\ 0 & 0 & 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & 0 & -1 & -1 & 3 & -1 \\ 0 & 0 & 0 & -1 & 0 & -1 & 2 \end{bmatrix}$$

We can get a good idea of the best way to partition a graph from the eigenvalues and eigenvectors of its Laplacian matrix. The smallest eigenvalues and their eigenvectors of a Laplacian matrix of a graph contain the information which is beneficial to divide the graph into two separate parts. Since the smallest eigenvalue for Laplacian matrices is every time 0 and its corresponding eigenvector is consisting of all ones, such that $[1, 1, \ldots, 1]$, Newman's spectral algorithm uses the second smallest eigenvalue of the Laplacian matrix. After that, the leading eigenvector of the second smallest eigenvalue is computed. The graph is divided into two subgraphs according to the signs of the elements in this vector, where nodes corresponding to negative elements of the vector are placed into one of these subgraphs and nodes corresponding to positive elements of the vector are placed into the other subgraph. By this way, negative and positive nodes are grouped together. This phase repeats recursively on newly created subgraphs until the leading eigenvector consists of the same signed values. Many networks, on the other hand, contain more than two communities, so the method needs to be extended to find better defined divisions of networks into larger numbers of parts. In our study, this method is used as a preprocess step in specific situations.

Genetic algorithm is a kind of optimization algorithm which imitates the genetic science and natural selection [30, 31]. In real world, individuals cross over their genes in which their genetic data are hold, and generate new offsprings. Also, sometimes, a gene of an offspring can be mutated. If the crossovered and mutated chromosomes of the offspring have good genetic data to adapt the environment, then the offspring will survive. Individuals who cannot adapt the environment will be extinct [32]. In genetic algorithms, better and high performing samplings can be constructed from the best partial solutions of the past samplings instead of trying every conceivable combination of which solution space consists. This is referred to as *the building block hypothesis* [31]. Genetic algorithms adopt these approaches and try to generate optimized solutions from a given initial set of solutions by using operators of mutation, crossover, and selection. A *fitness function* is defined to evaluate the fitness of the individual for the environment or the solution, and a score is produced as an output of the fitness function. After the evaluation phase, offsprings having the best fitness score survive for next generations. Steps of a traditional genetic algorithm are as follows:

1. Initially, fixed number of chromosomes are generated. Number of the chromosomes are called as *population* and it is given in the beginning.
2. New chromosomes are generated by *crossover*.
3. If a given probability occur, some chromosomes are affected from a *mutation*.
4. Genetic algorithm produces new chromosomes. Each of the chromosomes is evaluated through a cost function which is always referred to as *fitness function*.
5. Chromosomes with better fitness score are replaced with worst ones because the population needs to be fixed. This step is called as *selection*.
6. If aimed fitness score is reached, then stop else go to step 2.

Crossover is the core process of the genetic algorithm. Two chromosomes from the solution space are chosen randomly and one crossover point or crossover points, depending on the crossover type we choose, is determined on each chromosome. In one-point crossover, one point on each chromosome is selected and chromosomes are divided into two pieces. Pieces with same location and size of the chromosomes are exchanged within them and new offsprings are generated, as seen in Fig. 2. In two-point crossover, chromosomes are cut from two points and middle pieces of the chromosomes are exchanged, as seen in Fig. 3. In uniform crossover, some genes of the one parent chromosome are chosen which is also referred to as crossover bits, and these crossover bits are exchanged with other parents genes, as seen in Fig. 4.

Mutation is a probabilistic operation. If a predefined probability occurs, then mutation on a randomly selected chromosome occurs and values of the mutated

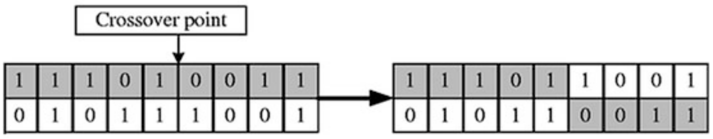


Fig. 2 One-point crossover in genetic algorithms

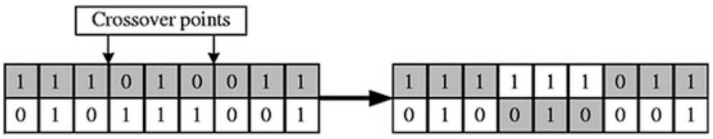


Fig. 3 Two-point crossover in genetic algorithms

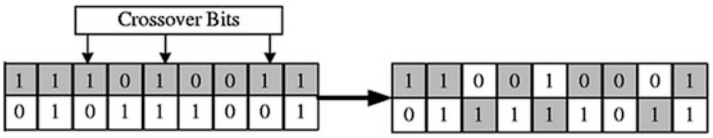


Fig. 4 Uniform crossover in genetic algorithms

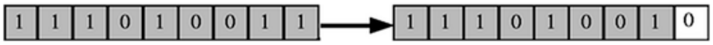


Fig. 5 Mutation in genetic algorithms

genes change. Mutation might be on a single random gene or might be on several random genes. In Fig. 5, a mutation example is shown.

Inversion operator is not always applied on real datasets. It does not change the information on the genes, it changes the presentation of the information on these genes.

In the work of Tasgin et al. [33, 34], detecting community structures in social networks is aimed by the help of genetic algorithms. They presented a genetic algorithm for detecting community structures in a network, referred to as GACD, by maximizing network modularity score. In GACD, a chromosome is encoded as a list or an array and a gene is encoded as an element on the list. Every node in the network is presented as a gene so the length of the list equals to the number of the nodes in the network. Values of the elements in the list present community number of equivalent nodes. In this encoding, places of the nodes on the list are constant, while value of the elements which are holding community numbers can change during the run of the algorithm. In initialization phase, all nodes in the network assigned to a community randomly. Since number of the genes equals to number of the nodes in the network, number of the communities is limited to the number of nodes, n. Each node may be assigned to a separate community in the worst case, and there will be n communities in the network in such a case. Also, during the initialization, a predefined variable called randomization rate is defined that random nodes are chosen according to this randomization rate and if the randomization rate is provided, then the neighbors of the randomly chosen node are assigned the same community number.

During the *crossover* stage of GACD, genes are not exchanged simply as in traditional genetic algorithm, instead the community identifiers of the nodes in a chromosome are moved to nodes in the destination chromosome. It should be mentioned that because of community numbers' being assigned randomly in each chromosome and not having a relation between them in the beginning, different community numbers in different chromosomes may mean the same community. For example, community number 1 in chromosome A and community number 34 in chromosome B might own nodes, on the other hand, community number 1 in chromosome B might not have a common node with community number 1 in chromosome A. To eliminate the disadvantage of this encoding, one way crossover is implemented in GACD. For *mutation*, a node is transferred into a different community randomly in the network. A random node is chosen and a new community number is generated by modifying a digit in its binary representation.

Furthermore, a novel operator named *Clean-up* is introduced by Tasgin et al. [33, 34] which is based on a new metric called *community variance* intending to reduce misplacement of the community numbers caused by encoding of the chromosomes. *Clean-up* is added as a next step after crossover and mutation. If the number of such misplacement is high, it is detected by the mechanisms of genetic algorithm via fitness evaluation. However, although the overall fitness value is good for a community split, there may be a small number of misplaced nodes that does not affect the overall fitness value very much. Although *Clean-up* step produces accurate results, the experiments are done only on small datasets[33, 34] like Zachary's Karate Club Network [35] and American College Football Network [11].

3 Methodology

In this section, our method for community detection in social networks based on genetic algorithms is introduced. For large real world datasets, a spectral method proposed by Newman [36] is used as a preprocess step for the initialization stage of the genetic algorithm. Each node of a network is accepted as a member of sole community, meaning that overlapping communities [37, 38] are not considered. Next subsections give detailed information about the steps of our algorithm, its encoding, operators, fitness function, and selection.

3.1 The Algorithm

We first define the following variables:

- N is the population size.
- p_{cr} is the probability parameter for the randomness to reinsert nodes after a crossover operation. If a randomly generated number is less than the p_{cr}, then idle nodes are placed into communities randomly, else they are inserted to communities in which they will have more neighbors.
- p_m is the probability for the mutation. If a randomly generated number is less than p_m, then mutation is applied.
- p_{mr} is the probability for the randomness to reinsert nodes after mutation. If a randomly generated number is less than the p_{mr}, then idle nodes are placed into communities randomly, else they are inserted to a community where they have more neighbors.
- Q_F is the desired modularity score. If it is reached, the algorithm terminates.
- I_F is the number of iterations to terminate the program, it creates an upper limit for the iteration count.

Steps of the algorithm to detect communities in social networks are as follows:

1. Preprocess: An optional step that a selected community detection algorithm is run on the large dataset up to predefined number of iterations to generate a good initial pool.

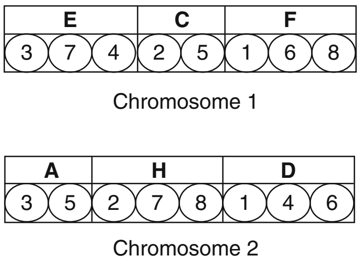
2. Initialization: If a preprocess step is applied, then community structures that are output of the preprocess step are used as input for the initialization. Nodes that are not assigned to a community after preprocess are assigned to newly created communities which can be very large. Input values do not constrain the algorithm since community structures evolve and number of communities can change through the run of the algorithm. If a preprocess is not applied, then each node is assigned to a community number randomly and community structures are created for the nodes assigned to these community numbers.
3. Evaluation: Evaluate the fitness value of all chromosomes.
4. Selection: Choose first N chromosomes with highest modularity score to survive for the next steps. Remove the others from the solution space.
5. Crossover: Crossover is applied $N/2$ times to randomly selected pair of chromosomes. For the probability of p_{cr}, reinsert idle nodes to communities randomly, else reinsert them to communities where they have largest number of neighbors.
6. Mutation: Mutation is applied to each chromosome with a probability of p_m. For the probability of p_{mr}, reinsert idle nodes to communities randomly, else reinsert them to communities where they have largest number of neighbors.
7. Control: If the best fitness value (i.e., modularity value) reaches to Q_F or iteration count reaches to I_F, then program terminates, else proceed to step 3.

3.2 Encoding and Initialization

Encoding of the genetic algorithm is inspired by the grouping genetic algorithms (GGA) which is proposed by Falkenauer [39]. Our focus is shifted from individuals of the network to communities of the network. In our encoding, communities are represented as the genes of the chromosomes. Every gene in the chromosome represents a community. The data that is held in the genes contains nodes belonging to that community. During crossover and mutation processes, nodes can change their communities. Figure 6 contains an example for encoding of two chromosomes. Chromosome 1 consists of three communities which are shown as genes, and the data held in genes contains their nodes. In chromosome 1, nodes 3, 7, and 4 belong to community E, 2 and 4 belong to community C, and 1, 6, and 8 belong to community F. Chromosome 2 is also similar, community A has nodes 3 and 5, community H has 2, 7, and 8, and community D has 1, 4, and 6.

In the initialization step, communities are generated and the nodes are assigned. If we use preprocessing step, communities coming from the preprocessing step are generated and the nodes belonging to these communities are assigned. For each node that does not belong to a community, a new community is generated and the node is assigned to that community. The preprocessing step can be skipped when size of the network is small: n communities are generated where n is the number of nodes in the network and each node is assigned to a sole community so that no more than one node is assigned to the same community.

Fig. 6 Example of encoding



3.3 *Fitness Function*

The network modularity metric (Eq. (1)) proposed by Newman et al. [29] and introduced in Section 2 is used as fitness function in our work, since it is a wide used way to evaluate the quality of community structures in networks. Also, network modularity is applied by Tasgin et al. [33] to their genetic algorithm as fitness function.

Concerning community i, let m_i be the total number of edges that have at least one side inside the community and let E be the number of edges in the network. Initial value of m_i is set to zero and we calculate m_i as follows: each edge incident with two nodes/one node belonging to community i increases m_i by 1/0.5, respectively. Finally, we compute m_i/E, and find a_i. We find the modularity scores Q_i for each community i and their summation gives as the modularity score of the network, Q as given in Eq. (2).

$$Q = \sum_i Q_i \tag{2}$$

After each iteration, the fitness score of each chromosome is recalculated and the chromosomes having higher modularity score are advantageous for survival at the next iteration.

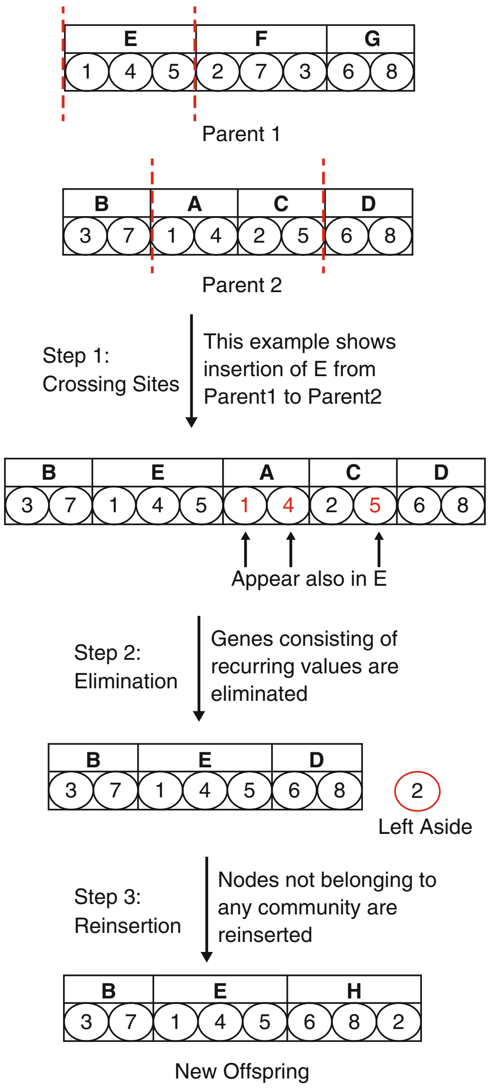
3.4 *Crossover*

Crossover is done between two randomly selected chromosomes and an offspring chromosome is produced. Our crossover method differentiates from traditional crossover methods: Nodes will be exchanged between the communities. With two parents it is possible to create two children by inserting the selected communities of the first parent into the second one, and by doing the opposite (Fig. 7).

The crossover consists of four steps:

1. Two crossover points are selected on each parent randomly and the parts between these points are chosen as crossing sites.

Fig. 7 Crossover



2. The communities selected by the crossing site of one parent are inserted at the crossing site of the second parent. At this stage, some nodes may appear in more than one community.
3. The communities which already exist in the second parent and contain nodes that are already in the inserted communities are eliminated, for example, some nodes do not belong to any community anymore. If some communities become empty after this elimination process, they are also removed from the solution.

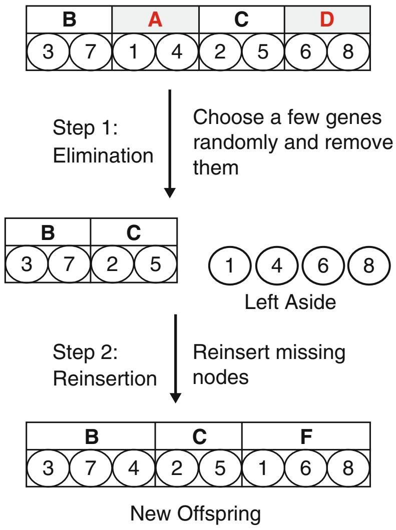
4. The nodes left aside are reinserted into the solution. According to a predefined probability rate, the node is put in a community randomly, or put in a community in which it will increase the network modularity of that community the most or decrease it least.

3.5 Mutation

The role of a mutation operator is to insert new characteristics into a population to enhance the search space of the genetic algorithm. In the case of our method, randomly a few communities are selected according to a predefined probability rate and the selected communities are eliminated from the chromosome. After the elimination, there will be nodes which do not belong to any community. The nodes are put in a community randomly or put in a community in which it will increase the modularity score of that community. If a predefined random value for node reinsertion is met, then the node will be put in a community to increase the modularity score of that community. On the other hand, if the predefined random value is not met or there are not any community that increase its modularity score, then the node put in a community randomly, or put in a newly created community.

In Fig. 8, mutation step is exemplified: genes A and D are selected randomly from the chromosome and they are eliminated. Nodes in communities A and D are reassigned to other communities. With the probability p_1, nodes put in a community in which they have more neighbors and if we cannot find a community in which node has neighbors, we generate a new community with the probability p_2 or put in a community randomly with the probability $1 - p_2$. With the probability $1 - p_1$, they are placed in a community randomly.

Fig. 8 Mutation



3.6 Selection

After the generation of new offsprings which are produced by crossover and mutation operators to the chromosomes, chromosome number exceeds the predefined population. Chromosomes must be chosen at the number of predefined population for the next iteration. First, fitness value of all chromosomes and newly generated offsprings are evaluated, and then they are sorted in decreasing order according to their fitness values. We take the chromosomes with best fitness values and place them in the population. This selection can be referred to as *elite selection*, only selecting the chromosomes with best fitness values.

3.7 Preprocess

We realize that our algorithm performs well if the number of nodes is not larger than 1000. However, for large real world dataset, chromosomes in the initial population make a significant difference. Starting the algorithm with the good community structures and doing operations on these well-defined structures help the evolving of the community structures of the network rapidly and accurately, as it is mentioned in the *building block hypothesis* [15, 31]. Therefore, we decided to use an algorithm to build a good initial pool in order to generate good results, which means producing higher network modularity scores, for detecting communities in a network. We use several of the communities produced from these algorithms as building block inputs.

Spectral algorithm for partitioning graph of a social network [29] and the Girvan–Newman algorithm [11] are used as a preprocessing step. These methods are run for limited number of steps. The network modularity functions used by these algorithms and our specialized genetic algorithm are the same. Communities generated by this step are used as initial communities for the genetic algorithm. We do not use all the community structures produced by them. If the size of the community is bigger than one third of the network size, it is not used as an input because it can dominate the result. Also, when getting inputs, overlapping communities are not used, every node must be in a sole community.

4 Experimental Results and Discussion

We used well-known datasets given in Table 1 to evaluate performance of our algorithm and existing ones.

While defining the parameters for each method for comparison, we get the parameters which are giving the maximum modularity scores. After the replacement of the optimal values of the parameters we found for each algorithm, we run each of

Table 1 Datasets used in experiments

Network	Size	Cited from
Zachary's Karate Club	34	[35]
Collaboration in Jazz	198	[20]
Metabolic	453	[22]
E-mail	1133	[21]
Facebook(NIPS) data	2888	[40]
PGP	10,680	[41]
Cond-Mat	27,519	[42]

them 50 times and take their highest modularity Q value for comparison. Each run is done for 500 iterations to let them reach their peak points.

4.1 Datasets

In this section, the datasets that are used through the experiments are introduced and their main characteristics are explained. *Zachary's Karate Club* [35] is a very popular network used by many researchers. It shows a university karate club which is divided into two communities after a conflict. The network consists of 34 nodes and 78 edges.

Collaboration in Jazz network [20] dataset was obtained from The Red Hot Jazz Archive digital database. The data consists of 198 bands that performed between 1912 and 1940, that most of them were seen performing in the 1920s. Each node in this dataset matches to a band, and an edge between two bands exists if at least one musician exists who played at both bands in any time. The network consists of 198 nodes and 2742 edges. In Fig. 9, outcome of our modified genetic algorithm presented in this paper is shown that it separates *Collaboration in Jazz network* into three communities. Nodes belonging to the same community colored with the same color.

emphMetabolic network [22] is the graph that showing the metabolic pathways of a multicellular organism, Elegans [3]. This network consists of 453 nodes and 2032 edges.

E-mail network dataset [21] is the network of e-mail interchanges between the members of the University of Rovira i Virgili in Tarragona. This network consists of 1133 nodes and 5451 edges.

Facebook(NIPS) dataset [40] It has the data of the Facebook users who installed the application of a Facebook Social API. This undirected network contains Facebook user to user friendships. A node represents a user. This network consists of 2888 nodes and 2981 edges.

PGP network [41] is the graph of a component of a network, consisting the users of the pretty-good-privacy algorithm for secure information interchange. PGP network contains 10,680 nodes and 24,316 edges.

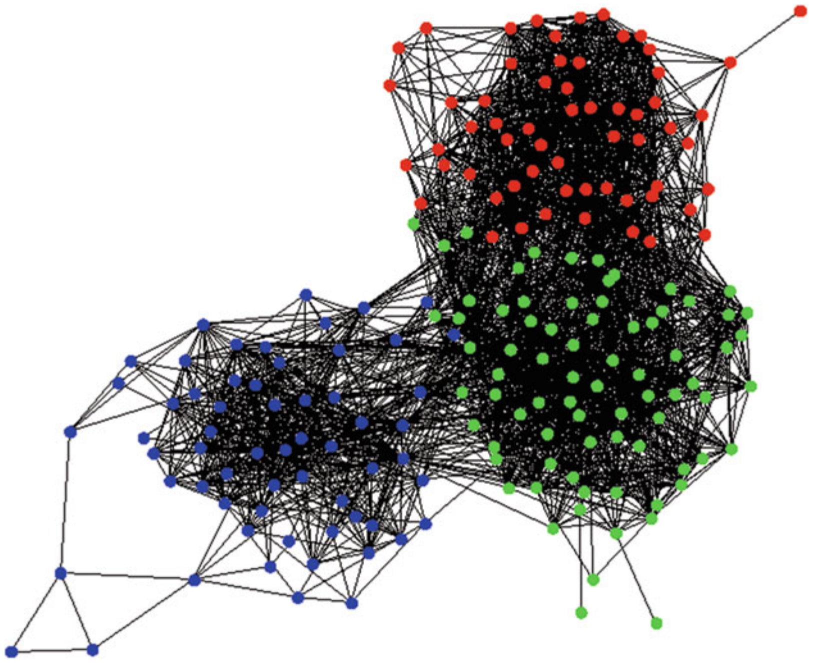


Fig. 9 Communities of *Collaboration in Jazz Social network* found by our method

Cond-Mat network [42] shows the relationships between the authors that shared any paper on Condense Matters. Cond-Mat network consists of 27,519 nodes.

4.2 Comparison Between Genetic Algorithms

Comparison between genetic algorithms is done within three different algorithms. First one is the traditional genetic algorithm, referred to as TGA, is a plain genetic algorithm that uses modularity for fitness function. Second one is the algorithm of Tasgin et al, referred to as GACD, and the last one is our modified genetic algorithm proposed in this work, referred to as MGA.

As you can see in Table 2, MGA gives accurate and satisfactory results when modularity scores compared to TGA and GACD. Especially, significant differences between maximum modularity scores reached on *e-mail* and *PGP* datasets can be seen.

In the results shown in Fig. 10, you can see how modularity scores evolve over each iteration on the Zachary's Karate Club dataset. Modularity scores are shown for every 10 turns. As you can see from the graphics, modularity scores obtained

Table 2 Comparison of modularity scores of GA technics

Network	TGA	GACD	MGA
Zachary's Karate Club	0.4198	0.4198	0.4198
Collaboration in Jazz	0.4393	0.4444	0.4444
Metabolic	0.3341	0.4338	0.4373
E-mail	0.2555	0.4339	0.5654
Facebook(NIPS) data	0.7980	0.8086	0.8087
PGP	0.7893	0.8071	0.8557

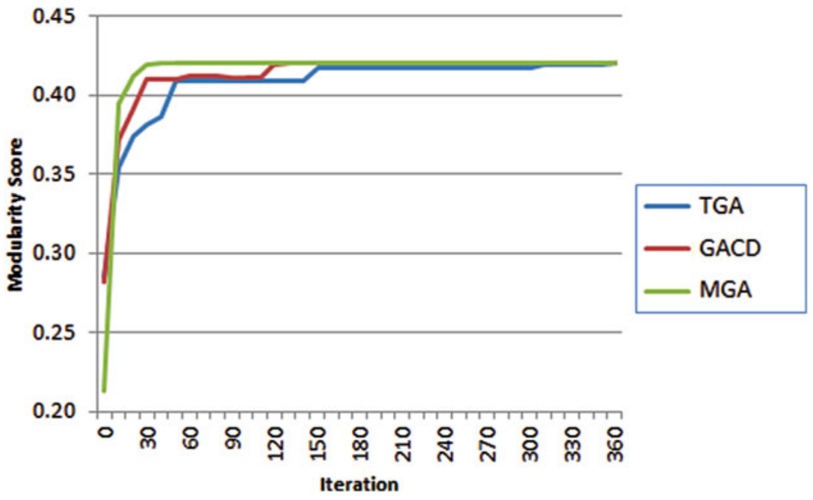


Fig. 10 Comparison on Zachary Karate Club over iterations

from MGA evolve better and reach higher values quickly although GACD and TGA have a good start.

In Fig. 11, comparison is made on the dataset of *Collaboration in Jazz Social network*. Modularity scores are again shown for every 10 turns. Only after 20 turns, our method reaches the peak modularity score while GACD needs around 80 turns for the same performance. Also, in Fig. 12, evolution of the modularity scores of all algorithms is plotted for every 5 s. Our algorithm approaches the peak point in about 5 s while GACD needs 10 more seconds and TGA needs 130 more seconds to catch up.

In Fig. 13, comparison is made on the *metabolic network*. The graph shows the runs of the algorithms up to maximum modularity scores. Modularity score values are taken from each iteration. Also, Fig. 14 shows the modularity score values taken for each 5 s. As you can see, there are not any important difference between GACD and MGA on *metabolic network* dataset.

In Fig. 15, comparison is made on the *e-mail network* dataset that each of the algorithms is run with their optimal parameters. The figure depicts the values of the modularity score Q for every 10 iterations. As you can see, all three algorithms start from a point which is very close to each other. MGA evolves better and approaches

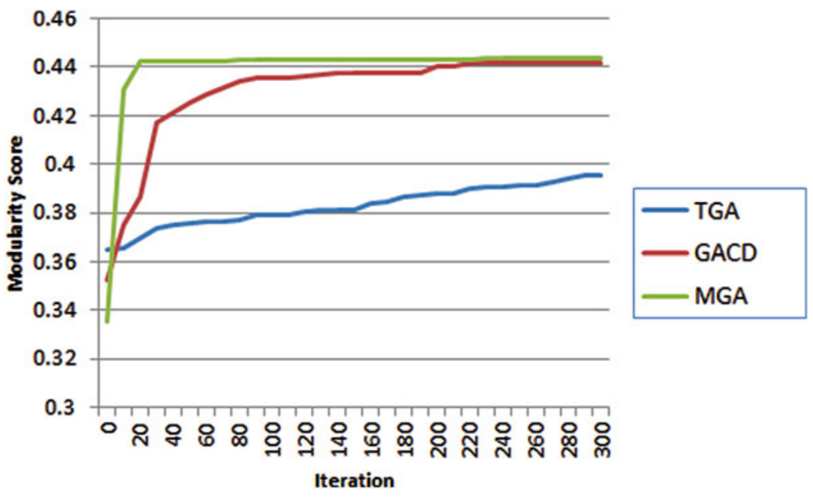


Fig. 11 Comparison on Collaboration in Jazz over iterations

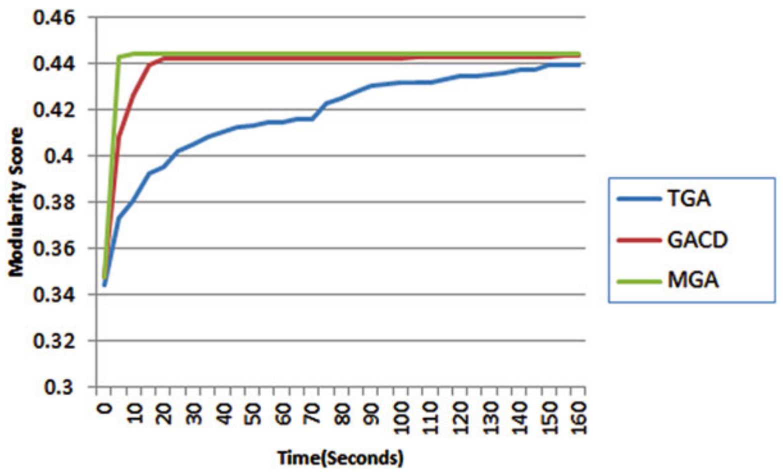


Fig. 12 Comparison on Collaboration in Jazz over time

to peak point quicker than the other algorithms. It can also be seen from the graph that when our algorithm reaches 0.4940, the highest modularity Q value for this dataset, at the iteration 300, modularity score of the TGA is around 0.26 and that of the work of GACD is around 0.42. In Fig. 16, we run these three algorithms 50 times and use the values from their best results to compare their execution times. We plot the evolution of the modularity scores for every 5 s. As you can see, MGA takes a large step after fifth second while TGA and GACD perform poorly. At 200, our algorithm reaches the value of 0.4940 while modularity score of the TGA is around 0.24 and that of GACD is around 0.32.

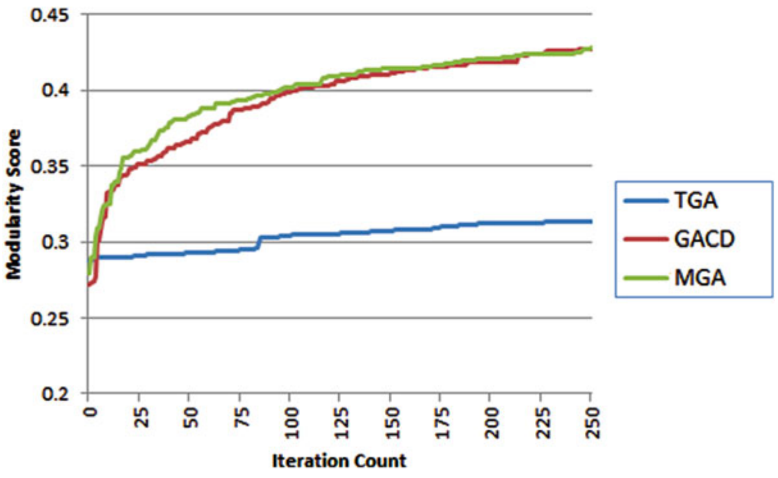


Fig. 13 Comparison on metabolic over iterations

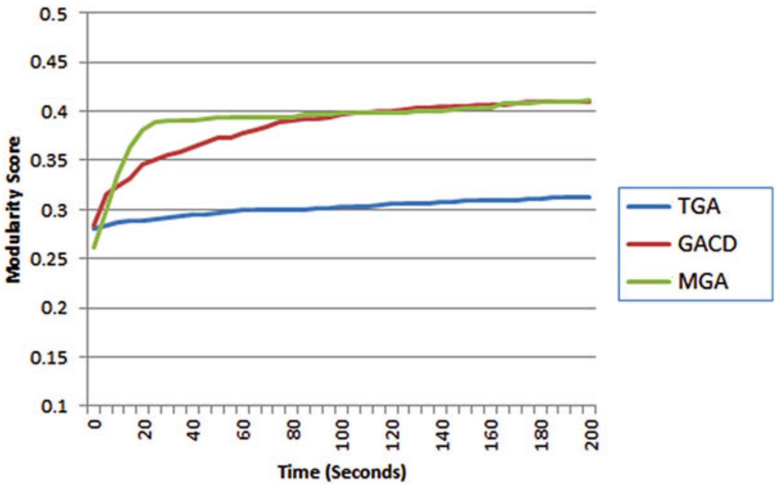


Fig. 14 Comparison on metabolic over time

In Fig. 17, comparison is made on the *Facebook (NIPS) network*. The graph shows the runs of the algorithms up to maximum modularity scores. Modularity score values are taken from each iteration. Also, Fig. 18 shows the modularity score values taken for each 5 s. After the initialization phase, algorithms begin with high modularity scores because of the structure of the dataset, then MGA reaches the value 0.8087 earlier.

In Fig. 19, comparison is made on the *PGP network* which has 10,680 nodes and is larger than the previously used datasets in terms of size. The graph shows the best performing runs of each of three algorithms throughout the iterations. Modularity

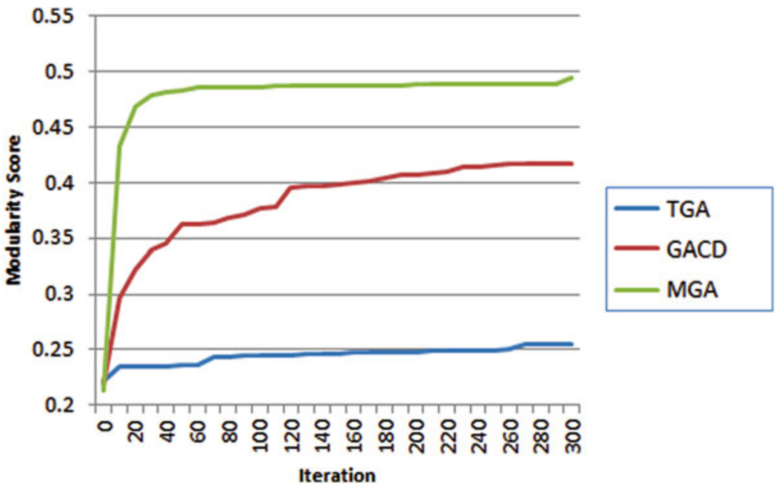


Fig. 15 Comparison on e-mail network over iterations

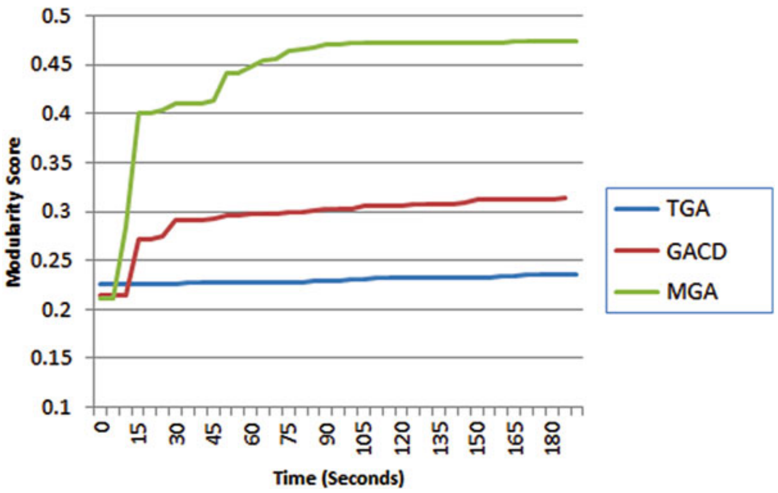


Fig. 16 Comparison on e-mail network over time

score values are taken from every 10 iterations. After the initialization phase, it is seen that all three algorithms have close modularity scores. After that MGA reaches the higher values at earlier iterations. In Fig. 20, we see how modularity scores of the algorithms evolve over the time on the *PGP network*. When we looked at the modularity scores obtained in first 200 min, it is seen that MGA performs better when compared to TGA and GACD.

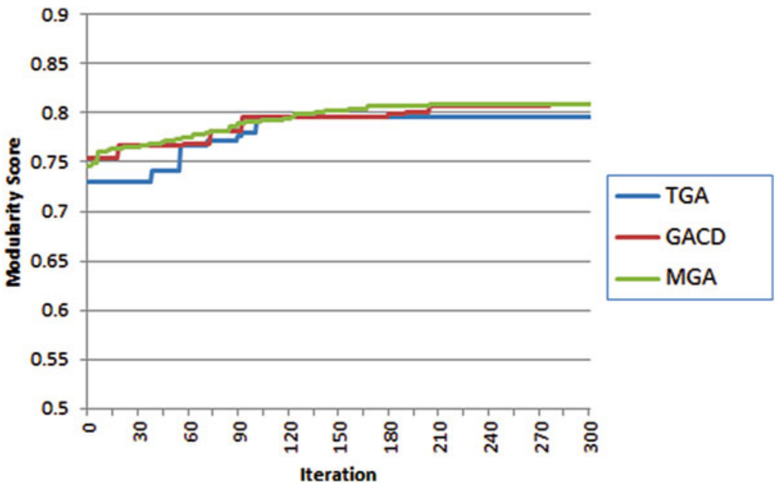


Fig. 17 Comparison on Facebook (NIPS) dataset over iterations

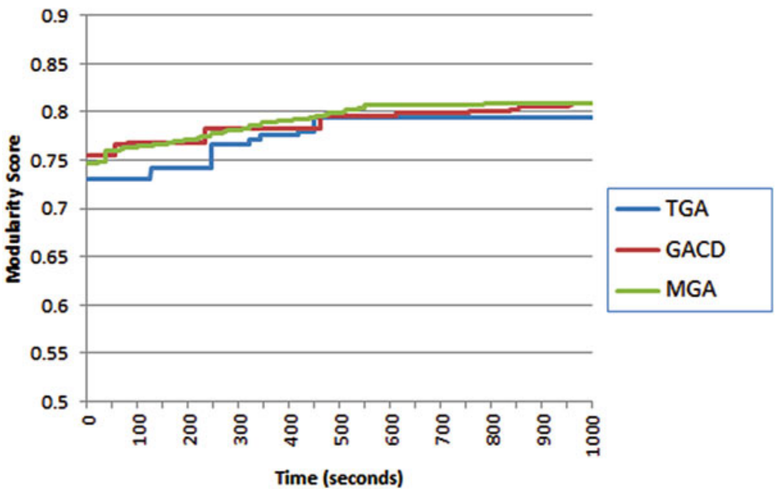


Fig. 18 Comparison on Facebook (NIPS) dataset over time

4.3 Comparison with Different Community Detection Algorithms for Social Networks

We compare our genetic algorithm with other four algorithms which are accepted as proven ways for community detection in social networks. These four algorithms are the following: the betweenness based algorithm of Girvan and Newman (abbreviated as GNA) [11], the greedy optimization algorithm of Clauset et. al. (abbreviated as CNM) [23], the extremal optimization algorithm of Duch and

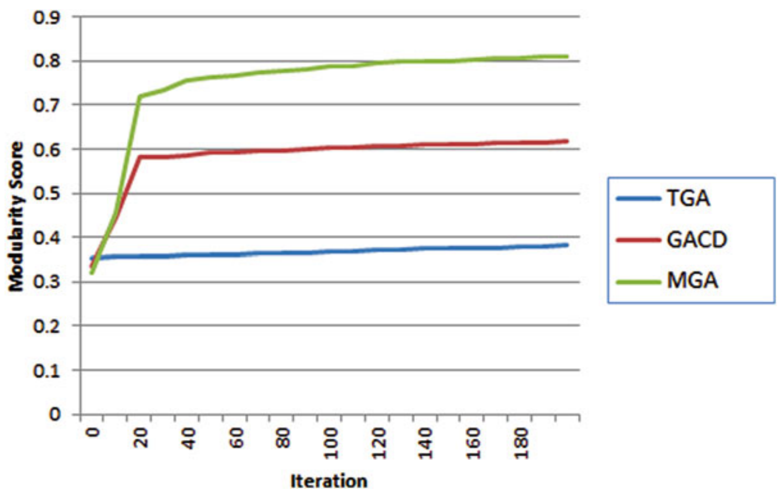


Fig. 19 Comparison on PGP dataset over iterations

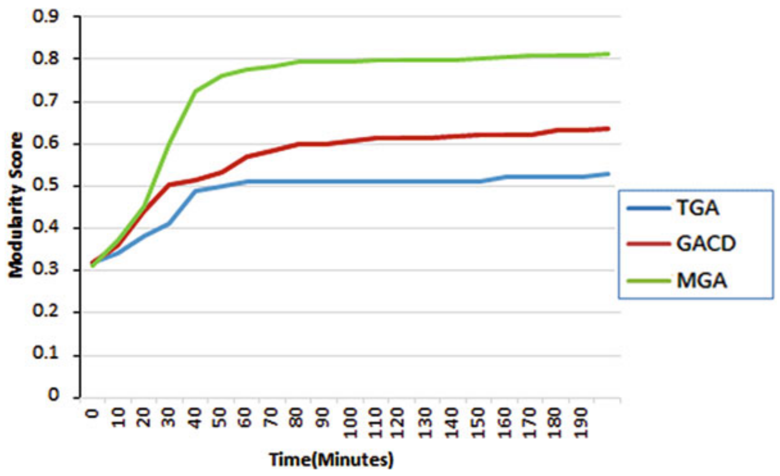


Fig. 20 Comparison on PGP dataset over time

Arenas (abbreviated as DAA) [22], and the spectral method of Newman (abbreviated as CNM) which uses eigenvectors of similarity matrices of the networks [29].

The results of the algorithm of Duch and Arenas [22] and algorithm of Clauset et al. [23] are taken from the experiments mentioned in other studies [22, 29]. For Girvan–Newman algorithm, the software *Gephi* (https://gephi.github.io/) and its *Girvan–Newman clustering plugin* is used. For the experiments for Newman's spectral algorithm, *Jmod Tool* (http://tschaffter.ch/projects/jmod/) which is a toolkit for community detection in networks is used. We did not need to find parameters for

Table 3 Comparison of modularity scores of MGA and other algorithms

Dataset	GNA	CNM	DAA	NSA	MGA
Zachary's Karate Club	0.401[a]	0.381[b]	0.419[b]	0.420[a]	0.420
Collaboration in Jazz	0.405[a]	0.439[b]	0.445[b]	0.442[a]	0.445
Metabolic	0.403[a]	0.402[b]	0.434[b]	0.424[a]	0.437
E-mail	0.532[a]	0.494[b]	0.574[b]	0.552[a]	0.565
PGP	0.816[a]	0.733[b]	0.846[b]	0.855[a]	0.856
Cond-Mat	Not applicable[b]	0.668[b]	0.679[b]	0.723[a]	0.723

[a]This data is not found by our experiment. It is claimed in and taken from the work of Newman
[b]This data is not found by our experiment. It is claimed in and taken from the work of Duch and Arenas

Table 4 Comparison of modularity scores of MGA with preprocess step and other algorithms

Dataset	GNA	CNM	DAA	NSA	MGA + Preprocess
Zachary's Karate Club	0.401[a]	0.381[b]	0.419[b]	0.420[a]	Not applied
Collaboration in Jazz	0.405[a]	0.439[b]	0.445[b]	0.442[a]	Not applied
Metabolic	0.403[a]	0.402[b]	0.434[b]	0.424[a]	0.438
E-mail	0.532[a]	0.494[b]	0.574[b]	0.552[a]	0.576
PGP	0.816[a]	0.733[b]	0.846[b]	0.855[a]	0.858
Cond-Mat	Not applicable[b]	0.668[b]	0.679[b]	0.723[a]	0.729

[a]This data is not found by our experiment. It is claimed in and taken from the work of Newman
[b]This data is not found by our experiment. It is claimed in and taken from the work of Duch and Arenas

other algorithms since experiments are done for these algorithms on datasets we are using and their best results are claimed in [11, 22, 23, 29].

Algorithms are run on the same datasets and their modularity scores are calculated. In Tables 3 and 4, you can see the comparison of modularity scores of these algorithms. Datasets are, in order, Zachary's Karate Club [35], Collaboration in Jazz network [20], metabolic network for the nematode C. Elegans, e-mail network [21], PGP which is a trust network on mutual signing of cryptography keys, and Cond-Mat which is a network showing the relations of the authors and the papers in Condense Matters archive.

For *metabolic network* mutation rate is taken as 0.8, initial chromosome count is taken as 100. p_{cr} is taken as 0.7, p_{mr} is taken as 0.8. For *Cond-Mat network*, mutation rate is taken as 0.7, initial chromosome count is taken as 100. p_{cr} is taken as 0.7, p_{mr} is taken as 0.6. For both PGP and Cond-Mat, no maximum iteration count is defined. PGP and Cond-Mat are terminated if the modularity score does not change for past 100 and 200 iterations, respectively.

In Table 3, maximum network modularity scores of the algorithms are shown. Each algorithm is run on the same datasets 40 times and their largest modularity scores are reported. GNA is not applied to Condense Matters dataset because of its huge time requirement.

4.4 Experiments with Using Preprocess Step

In Table 4, also, maximum network modularity scores of the algorithms when they run on the same datasets are shown. Again, GNA is not applied for Condense Matters dataset. For this time, MGA is run including the preprocess step. We did not run MGA on Zachary's Karate Club and Collaboration in Jazz network datasets, because their size is too small for a preprocess step and it does not refine the results. For the preprocess step, we have taken communities from the results of GNA and NSA and created a pool consisting of communities. For metabolic and e-mail datasets, all the communities given as output after the run of GNA and NSA and communities not larger than one third of the network size are used as input. For PGP and Cond-Mat datasets, GNA and NSA are run for 1 h and the indivisible communities found during the first 1 h are used as input. Because of GNA is not being applied to Condense Matters dataset, we use only results from NSA.

In Fig. 21, time comparison of the modularity scores between NSA and MGA with preprocess is made. MGA has taken 15 different communities randomly assigned to the communities from the community pool generated by running NSA and GACD of Tasgin et al. for 1 h. The snapshots of the PGP network are taken when the NSA divided it into smaller pieces. Modularity scores of the NSA and MGA with preprocess are compared at the times when the mentioned snapshots are taken. Our pool of communities consists of 37 communities. Quality of these communities is determined by their modularity scores. It can be seen that modularity score of MGA gets larger in a fast way until it reaches a point, after that its move gets slower. Also, in Fig. 21, it is seen that MGA converges to a peak modularity score in a very short period of time compared to NSA.

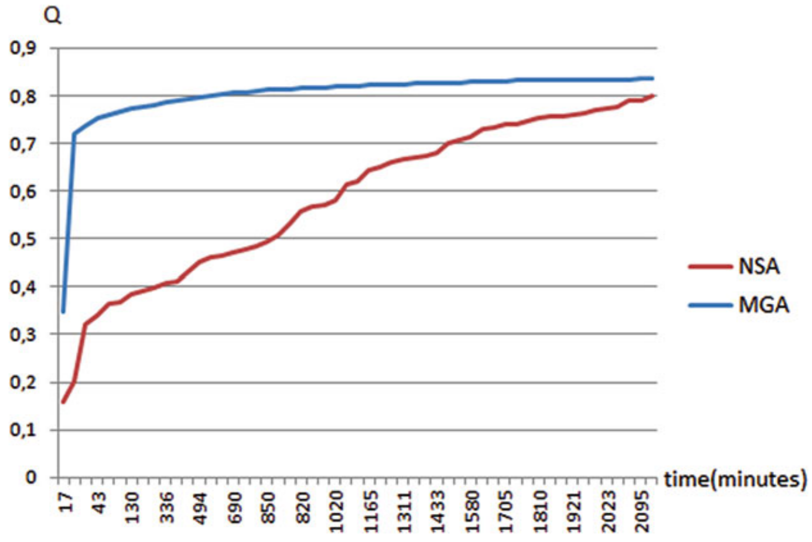


Fig. 21 Time comparison on PGP network

5 Conclusion

In this paper, requirements and methodologies for detecting communities in social networks are analyzed and a new method is proposed. Previous studies concentrating on the issues such as clustering, graph partitioning, and community detection in networks and also genetic algorithms are reviewed. Eventually, a method which benefits from genetic algorithms and specialized to detect communities in social networks is proposed. In this method, we shifted our focus, which is inspired from Falkenauer's presentation of grouping genetic algorithms, from individuals of the graph to communities of the graph. An encoding which presents communities as genes of the chromosomes and individuals as the data of the genes is applied. Additionally, for large real network data, a preprocess step which makes use of the community structure outputs produced from running of other algorithms is used. With the preprocess step, we planned to take advantage of building blocks for a more rapid run and accurate results. To evaluate the success of the algorithm, modularity score is used as fitness function.

First, our modified genetic algorithm is compared with other genetic algorithms: Traditional genetic algorithm and grouping genetic algorithm of Tasgin et al. The experiments have been done without applying a preprocess step to distinguish performances of genetic algorithms and to show how our encoding of chromosomes and operators perform. The outcomes were satisfactory such that our method finds community structures faster and reaches the peak modularity score consuming less time than the other two methods. We compared how modularity scores changed through the iterations and changed through the time passes. Experiments done with both traditional genetic algorithm and GACD of Tasgin et al. prove that our modified genetic algorithm (MGA) is superior to them while detecting communities of social networks even without using a preprocess step. Then, our method is compared to other algorithms [11, 22, 23, 29] which are using different graph partitioning or community detection approaches, but using modularity score, common in evaluating the structure of the network. As it is seen in the results, our modified genetic algorithm (MGA) produces good results when compared to others. For the need of improving the modularity score of the network further, a preprocess step is applied. Because the preprocess step provides building blocks to the solution space, it enhanced the modularity scores of the networks. As a result our method is an effective way of using building blocks.

Since our method uses a kind of genetic algorithm, it reaches the optimal solution eventually as expected and also it has rapid refinements, although there are some disadvantages. There are points that can be referred to as saturation points where improvement of the modularity score gets slower when it gets closer to the peak points. In future studies, we will look for solutions to overcome the saturation points in a faster way. Furthermore, implementation of parallel programming to our method can speed up the running time.

References

1. Scott, J.: Social Network Analysis: A Handbook. SAGE Publications, London (2000)
2. Wasserman, S., Faust, K.: Social Network Analysis. Cambridge University Press, Cambridge (1994)
3. Albert, R., Barabasi, A.-L.: Statistical mechanics of complex networks. Rev. Mod. Phys. **74**(47) (2002). https://doi.org/10.1103/RevModPhys.74.47
4. Barrat, A., Barthelemy, M., Vespignani, A.: Dynamical Processes on Complex Networks. Cambridge University Press, Cambridge (2008)
5. Boccaletti, S., Latora, V., Moreno, Y., Chavez, M., Hwang, D.-U.: Complex networks: structure and dynamics. Phys. Rep. **424**, 175–308 (2006)
6. Dorogovtsev, S.N., Mendes, J.F.F.: Evolution of Networks: From Biological Nets to the Internet and WWW. Oxford University Press, Oxford (2003). https://doi.org/10.1063/1.1825279
7. Newman, M.E.J.: The structure and function of complex networks. SIAM Rev. **45**(2), 167–256 (2003)
8. Pastor-Satorras, R., Vespignani, A.: Evolution and Structure of the Internet: A Statistical Physics Approach, Cambridge University Press, New York (2004)
9. Erdos, P., Renyi, A.: On random graphs I. Publ. Math. Debr. **6**, 290–297 (1959)
10. Fortunato, S.: Community detection in graphs. Phys. Rep. **486**, 75–174 (2010)
11. Girvan, M., Newman, M.E.J.: Community structure in social and biological networks. PNAS **99**(12), 7821–7826 (2002)
12. Crane, D.: Invisible Colleges: Diffusion of Knowledge in Scientific Communities. University of Chicago Press, Chicago (1972)
13. Egghe, L., Rousseau, R.: Introduction to Informetrics. Elsevier, Amsterdam (1990)
14. Breiger, R.L., Boorman, S.A., Arabie, P.: An algorithm for clustering relational data with applications to social network analysis and comparison with multidimensional scaling. J. Math. Psychol. **12**(3), 328–383 (1975). https://doi.org/10.1016/0022-2496(75)90028-0
15. White, H.C., Boorman, S.A., Breiger, R.L.: Social structure from multiple networks. I. Blockmodels of roles and positions. Am. J. Sociol. **81**(4), 730–780 (1977)
16. Radicchi, F., Castellano, C., Cecconi, F., Loreto, V., Parisi, D.: Defining and identifying communities in networks. PNAS **101**(9), 2658–2663 (2004)
17. Girvan, M., Newman, M.E.J.: Finding and evaluating community structure in networks. Phys. Rev. E Stat. Nonlinear Soft Matter Phys. **69**(2 Pt 2), 026113 (2004)
18. Newman, M.E.J.: Mixing patterns in networks. Phys. Rev. E **67**, 026126 (2003). https://doi.org/10.1103/PhysRevE.67.026126
19. Holme, P., Huss, M., Jeong, H.: Subnetwork hierarchies of biochemical pathways. Bioinformatics **19**, 532–538 (2003)
20. Gleiser, P., Danon, L.: Community structure in Jazz. Adv. Complex Syst. **6**(4), 565–573 (2003)
21. Guimera, R., Danon, L., Diaz-Guilera, A., Giralt, F., Arenas, A.: Self-similar community structure in organisations. Phys. Rev. E **68**, 065103 (2003)
22. Duch, J., Arenas, A.: Community detection in complex networks using extremal optimization. Phys. Rev. E Stat. Nonlin. Soft. Matter. Phys. **72**(2), 027104 (2005)
23. Clauset, A., Newman, M.E.J., Moore, C.: Finding community structure in very large networks. Phys. Rev. E 70, 066111 (2004)
24. Fiedler, M.: Algebraic connectivity of graphs. Czechoslov. Math. J. **23**(2), 298–305 (1973). Institute of Mathematics, Academy of Sciences of the Czech Republic
25. Pothen, A., Simon, H.D., Liou, K.: Partitioning sparse matrices with eigenvectors of graphs. SIAM J. Matrix Anal. Appl. **11**(3), 430–452 (1990)
26. Kernighan, B.W., Lin, S.: An efficient heuristic procedure for partitioning graphs. Bell Sys. Tech. J. **49**(2), 291–308 (1970)
27. Blondel, V.D., Guillaume, J.-L., Lambiotte, R., Lefebvre, E.: Fast unfolding of communities in large networks. J. Stat. Mech: Theory Exp. **2008**(10) (2008)

28. Richardson, T., Mucha, P.J., Porter, M.A.: Spectral tripartitioning of networks. Phys. Rev. Lett. E **80**, 036111 (2009)
29. Newman, M.E.J.: Modularity and community structure in networks. PNAS **103**(23), 8577–8582 (2006)
30. Goldberg, D.E.: Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley, Boston (1989)
31. Holland, J.H.: Adaptation in Natural and Artificial Systems. University of Michigan Press, Ann Arbor (1975)
32. Darwin, C.: On the Origin of Species. John Murray, London (1859)
33. Tasgin, M., Bingol, H.: Community Detection in Complex Networks Using Genetic Algorithms. arXiv:cond-mat/0604419v1 (2006)
34. Tasgin, M., Herdagdelen, A., Bingol, H.: Community Detection in Complex Networks Using Genetic Algorithms. arXiv:0711.0491 (2007)
35. Zachary, W.W.: An information flow model for conflict and fission in small groups. J. Anthropol. Res. **33**(4), 452–473 (1977)
36. Newman, M.E.J.: Finding community structure in networks using the eigenvectors of matrices. Phys. Rev. E, **74**(23), 8577–8582 (2006)
37. Palla, G., Dernyi, I., Farkas, I., Vicsek, T.: Uncovering the overlapping community structure of complex networks in nature and society. Nature **435**(7043), 814–818 (2005)
38. Pissard, N., Assadi, H.: Detecting Overlapping Communities in Linear Time with PA Algorithm. arXiv:physics/0509254 (2005)
39. Falkenauer, E.: Genetic Algorithms and Grouping Problems. John Wiley, New York (1998)
40. McAuley, J., Leskovec, J.: Learning to Discover Social Circles in Ego Networks, NIPS. Curran Associates Inc., Nevada (2012)
41. Boguna, M., Pastor-Satorras, R., Diaz-Guilera, A., Arenas A.: Models of social networks based on social distance attachment. Phys. Rev. E **70**, 056122 (2004)
42. Newman, M.E.J.: Scientific collaboration networks. I. Network construction and fundamental results. Phys. Rev. E, **64**(1), 016131 (2001). American Physical Society

On Detecting Multidimensional Communities

Amani Chouchane, Oualid Boutemine, and Mohamed Bouguessa

Abstract The study of multidimensional networks has become an active field of research in the last few years. One of the most fundamental tasks is community detection where the aim is to find subsets of densely connected or highly interactive nodes. Community detection in multidimensional networks has particularly gained a lot of attention and a number of approaches have been proposed. Still, several aspects remain to be addressed in the current literature. In fact, besides being parameter-laden, the majority of the proposed approaches thus far lack an outlier detection mechanism and systematic procedures for explicit selection of the dimensions associated with the detected communities. To cope with these limitations, we introduce a novel principled approach named MCDA: multidimensional community detection algorithm. The proposed approach comprises two phases: (1) handling outliers and (2) mining multidimensional communities. The first phase of the algorithm is based on a probabilistic approach that exploits the beta mixture model to identify and eliminate outlier nodes from a network in a systematic way. The second phase adopts a local search mechanism which is inspired from the label propagation principle to detect communities. To this end, we design a novel propagation rule that exploits the most frequently used interaction dimensions among neighbors as an additional constraint for membership selections. The new propagation rule allows MCDA to automatically unfold the hidden communities in a multidimensional context. The detected communities are further processed for selection of relevant dimensions using an inter-class inertia-based procedure.

Keywords Clustering · Multigraphs · Multidimensional communities

A. Chouchane · O. Boutemine · M. Bouguessa (✉)
Department of Computer Science, University of Quebec at Montreal, Montreal, QC, Canada
e-mail: bouguessa.mohamed@uqam.ca

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_3

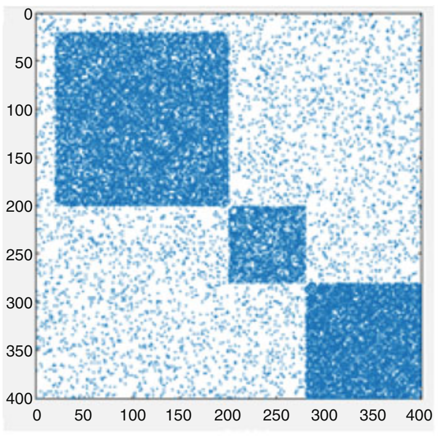
1 Introduction

Multidimensional networks have become an emerging model for representing complex interaction scenarios among entities of real world systems. A multidimensional network can be represented as a graph where each pair of connected nodes is linked by multiple edges that describe different types of interaction. This representation can be described by a set of elementary networks (i.e., dimensions) sharing the same set of nodes while capturing different kinds of interactions.

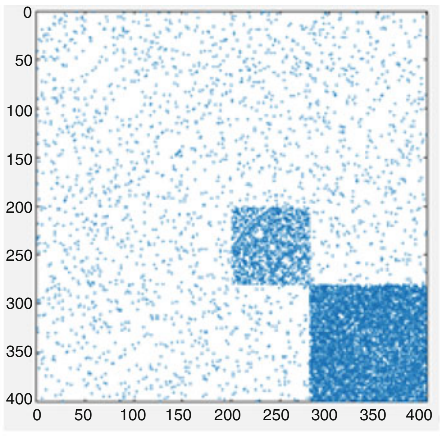
Recently, a number of research efforts have been devoted to the investigation of multidimensional networks. Some works were directed towards the characterization of their structural properties [1–3] while other proposals focused on the study of dynamic processes like percolation [4] and epidemic spreading [5]. In this paper, we focus on the problem of community detection. The goal is to automatically find subsets of densely connected nodes across different subspaces of dimensions. Nodes sharing membership to the same community are known to share common characteristics and behaviors. Thus, the discovery of communities within a network would give a better understanding of its structural features and hidden organization.

Community detection in multidimensional networks has received an increasing interest in the last few years and a number of techniques have been proposed. These techniques differ in the way they approach this problem and the definition they adopt for a multidimensional community. For instance, some methods, such as the work in [6], and in [7], consider a community as a set of densely connected nodes across all networks' dimensions. Accordingly, a community is supposed to exist in each one of the network's dimensions. However, such a definition appears to be less realistic since in multidimensional networks a community may also exist in different combinations of dimensions. This is why other proposals [8–10] regard a community as a region of high density of links that only exists along some dimensions. Such a definition implies the existence of a subset of dimensions supporting the formation of the community. This subset of dimensions is referred to as the relevant dimensions for the community. The remaining dimensions that do not contribute to the formation of community are called irrelevant dimensions. Here, it is important to note that multidimensional community detection algorithms are not limited solely to identifying community structures and their associated relevant dimensions. A multidimensional community detection algorithm should also be able to identify a set of outlier nodes which, by definition, do not lie within communities (in other words, outliers are those nodes that significantly deviate from densely connected nodes).

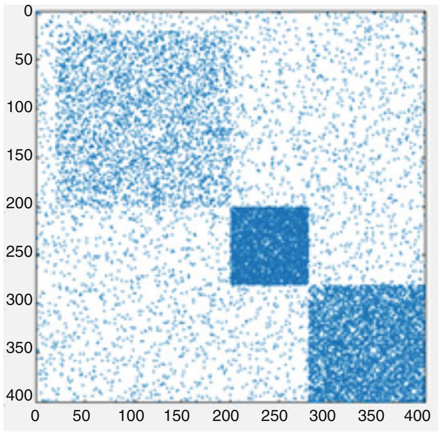
To illustrate, we generated a multidimensional network that contains four dimensions and three community structures embedded in different subspaces. Figure 1 illustrates the adjacency matrices of the four dimensions. In each matrix of this figure, shaded regions illustrate the presence of node-to-node connections along a specific dimension, while white regions denote the absence of such connections. Note that this four-dimensional network contains 400 nodes spanning 3 communities and 25 outlier nodes that are not located within any of the generated



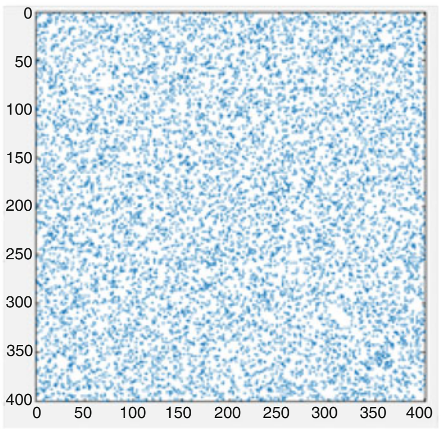
(a) d_1



(b) d_2



(c) d_3



(d) d_4

Fig. 1 An example of a four-dimensional synthetic network containing three communities across different combination of dimensions. Subfigures illustrate the associated adjacency matrices of each dimension. (**a**) d_1, (**b**) d_2, (**c**) d_3, (**d**) d_4

three communities. As can be seen from Fig. 1, dimensions d_1, d_2, and d_3 contain dense block regions describing three community structures projected along these dimensions. The first community exists in dimensions d_1 and d_3, while the second and the third communities exist in dimensions d_1, d_2, and d_3. On the other hand, dimension d_4 is completely an irrelevant dimension that does not exhibit any community structures. Outliers are those nodes that do not belong to any dense regions (communities) along all the dimensions of the generated network. Specifically, outliers are located at the top of each matrix of Fig. 1 in which no block (dense region) exists. In this paper, we are striving to (1) identify and eliminate such a type of outlier nodes since they harm the community detection process, and (2) elaborate a parameterless approach that can identify communities and their associated relevant dimensions in a fully automatic way. In what follows, we first describe briefly the mainstream algorithms for community detection in multidimensional networks. Next, we discuss a number of elements that motivate this study and describe our contributions.

1.1 Related Approaches

Thus far, the works addressing the problem of community detection in multidimensional networks have adopted various exploratory modes of the networks' structure. For instance, several methods [11, 12] attempt to reduce the problem of community detection to the classical setting by performing a transformation of the original multidimensional network into a monodimensional weighted graph. Community detection techniques for weighted graphs can thus be applied on the aggregated network. Recovering communities from individual dimensions and finding a consensus partition is another straightforward way to handle multidimensional networks. For example, the authors in [13] suggest the use of ensemble clustering strategies [14]. Berlingerio et al. [15] proposed an approach targeted to identifying overlaps between communities in the same or across different dimensions. Specifically, the work in [15] exploits a frequent closed itemsets mining algorithm [16] to process the list of nodes' memberships in the extracted communities. Nodes backing a frequent closed itemset capture, thus, a multidimensional community. Here, it is worth noting that consensus approaches are sensitive to sparse dimensions, that is, noisy dimensions that do not exhibit any community structures. Indeed, the high number of small communities recovered from these sparse dimensions might affect the consensus partitioning.

Following the previous model, feature integration-based methods consider the network's dimensions on an individual basis. However, unlike former approaches [13, 15] which combine the partitions obtained from each dimension, the most important structural features are extracted and combined instead. For instance, the PMM method (principal modularity maximization) [7] combines the spectral features of the modularity matrices of the network's dimensions. Singular value decomposition is then applied to produce a lower-dimensional embedding that serves as an input to a k-means algorithm. Several related feature integration schemes [17, 18] have also been proposed. For example, the authors in [17] proposed a linked matrix factorization-based approach whereas the work in [18] describes a spectral method named SC-ML (spectral clustering on multilayer graphs) for mining communities in multidimensional networks. Since they internally rely on k-means clustering, feature integration-based methods suffer from their dependency on the number of communities which should be specified by the user.

In order to handle multidimensional networks, several works have focused on the development of multidimensional alternatives from standard approaches.

For instance, a generalization of the well-known modularity measure was presented in [19]. The new definition was used by Carchiolo et al. [20] to design a Louvain-inspired [21] generalized modularity maximization approach. Likewise, De Domenico et al. [22] proposed a compression-based extension to the Infomap algorithm [23]. The information flow is modeled as a random walk where the best partition is recovered by minimizing a modified map equation. Multidimensional community discovery has also been investigated using tensor decomposition-based methods [24]. For instance, the work in [10] introduced GraphFuse, a clustering approach for multidimensional networks which uses a tensor factorization approach. Despite its ability to measure the relevance scores of the network's dimensions to the detected communities, GraphFuse's dependence on the targeted number of communities can limit its applicability.

Finally, it is worth noting that the output of some community detection algorithms, such as the ones proposed in [15, 19, 25], differs significantly from the type of the communities that we aim to discover in this paper. In fact, the approaches in [15, 19, 25] strive to identify all densely connected nodes in all subspaces of dimensions. These algorithms tend to uncover a very large number of communities since the same node may be assigned to multiple communities across multiple subspaces. Although, depending on the application domain, such an approach can be an interesting tool for mining multidimensional networks; the interpretation of their results poses a significant challenge due to the high number of detected communities. In this paper, we focus on algorithms designed to identify disjoint community structures which may exist in different combinations of dimensions. We believe that partitions identified by these algorithms provide clearer interpretability of results, as compared to reporting densely connected communities with high overlap.

1.2 Motivations and Contributions

The approaches described in the previous section do not provide systematic mechanisms to select the relevant dimensions associated with the detected communities. Besides, the majority of these approaches depend on a number of user-supplied parameters which require a proper tuning. These parameters can affect the detection accuracy if inappropriate values are provided. In addition, many algorithms require the number of communities to be fixed ahead of time. In real world applications, however, the ability to provide accurate values for these parameters is often limited since prior knowledge about the investigated network might not be available. In addition most existing approaches encounter difficulties when the network under investigation contains a high number of irrelevant dimensions compared to the number of relevant dimensions in which communities may exist. The presence of many irrelevant dimensions severely harms the community detection process of existing methods.

In [8], we proposed a parameterless approach which is able to automatically detect communities and their relevant dimensions. However, this approach is not able to handle outliers. The lack of a systematic mechanism to identify and eliminate outlier nodes limits the applicability of any community detection algorithm. In fact, while communities may hide in different subspaces, multidimensional networks are also characterized by the presence of outlier nodes that do not cluster well. Surprisingly, existing approaches designed to detect community structures in multidimensional networks do not include an outlier detection mechanism. Outliers are considered as noise that must be eliminated because they reduce community detection accuracy and impair the effectiveness of any mining techniques. We believe that multidimensional network applications require not only a partitioning algorithm to discover communities embedded in subspaces, but also an algorithm capable of handling outliers.

To cope with these limitations, we introduce an effective algorithm for community detection in multidimensional networks. The proposed approach (henceforth MCDA, short for multidimensional community detection algorithm) is able to automatically identify outliers and does not require any parameter setting to detect communities and their associated dimensions. MCDA is composed of two phases: outlier detection and community identification. In the first phase, starting from the assumption that outliers are inconsistent or considerably different from the remaining nodes in a network, we define a function to assess the degree to which a node is an outlier. Then, based on the estimated degree of outlierness, we devise a probabilistic approach to systematically discriminate outliers from nodes that may potentially lie within a community. In the second phase, MCDA adopts a propagation mechanism which exploits the most frequently used interaction dimensions among neighbors as an additional constraint for membership selection. These constraints are expressed through a weighting scheme which assigns to each neighbor, a score that represents the importance of the dimensions connecting the pair of nodes. These scores are then used to guide the propagation rule for the search of the best partition. Once the communities are recovered, our approach proceeds with the selection of their relevant dimensions. To this end, a relevance score is estimated for each dimension based on its contribution in links to the formation of the processed community. Relevant dimensions to a specific community will receive high score values, while irrelevant dimensions receive low relevance score. An inter-class inertia-based selection procedure is then applied to find the best cut position to separate the high scoring dimensions from the irrelevant dimensions with low score values. The following section provides a detailed description of our approach.

2 The MCDA Approach

2.1 Problem Statement

Multidimensional networks can formally be represented by multigraphs [3]. Let $G = (V, E, D)$ be an undirected and unweighted multigraph, where V is a set of n nodes; D is a set of o dimensions; E is a set of m edges, that is, the set of triplets (v, u, d) such that $v, u \in V$ are nodes, and $d \in D$ is a dimension. The triplet (v, u, d) specifies that the two nodes v and u are connected by one edge that belongs to dimension $d \in D$. Each pair of nodes in G can thus be connected by at most o possible edges.

In this paper, we assume that each node $v \in V$ belongs either to one community or to the set of outlier nodes OUT. A multidimensional community C_k ($k = 1, \ldots, K$), where K is an unknown number of communities, is defined as a pair (V_k, D_k) where V_k is a subset of $V' = V - OUT$; $D_k \subseteq D$ is a subset of dimensions such that nodes in V_k are densely connected across D_k. Dimensions in D_k are called relevant dimensions for the community C_k. The remaining dimensions of G, that is, $D - D_k$, are called the irrelevant dimensions for C_k. Note that the subsets of relevant dimensions $\{D_k\}_{k=1,\ldots,K}$ may or may not be disjoint and may have different cardinalities. A dimension can be relevant to zero, one, or more communities. To illustrate, consider the three-dimensional network depicted in Fig. 2. One would expect two communities to result from this multidimensional network: $C_1 = (V_1, D_1) = (\{n_1, n_2, n_3\}, \{d_1\})$ and $C_2 = (V_2, D_2) = (\{n_4, n_5, n_6, n_7\}, \{d_1, d_2\})$.

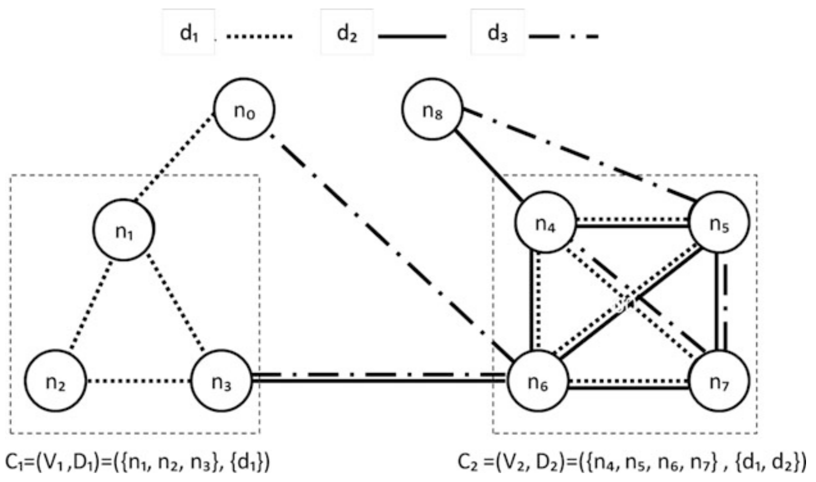

d₁
d₂
d₃
n₀
n₈
n₁
n₂
n₃
n₄
n₅
n₆
n₇
C₁=(V₁,D₁)=({n₁, n₂, n₃}, {d₁})
C₂ =(V₂, D₂)=({n₄, n₅, n₆, n₇}, {d₁, d₂})

Fig. 2 A three-dimensional network with two communities embedded along two different subsets of dimensions. Community C_1 exists in one dimension d_1 while community C_2 exists in dimensions d_1 and d_2

As can be seen, communities may exist in different combinations of dimensions. Nodes n_0 and n_8 are outliers since they exhibit atypical connections with respect to the rest of the network's nodes and they do not belong to any identifiable dense structure. Finally, note that a careful visual inspection of Fig. 2 suggests that dimension d_3 is an irrelevant dimension since it does not contain any meaningful structure in comparison to d_1 and d_2.

We devise in this section MCDA, a principled approach for automated discovery of multidimensional communities. To this end, our algorithm proceeds in two phases: (1) handling outliers, which aims to detect and eliminate outlier nodes that do not lie within any dense region in the network under investigation and (2) mining community structure, which aims to detect communities and their relevant dimensions. It is worth noting that the approach we propose is independent from any network-specific parameter such as the number of communities or a detection threshold to discriminate outliers from normal nodes that belong to communities. Details of each phase are given in the following.

2.2 *Outlier Handling*

In this phase, as mentioned previously, we focus on the problem of detecting outlier nodes that do not belong to any dense regions across all the network dimensions of G. To this end, we proceed in two steps. In the first phase, we investigate the network topology in order to estimate an outlier score for each node. Outlier nodes receive small scores, while nodes that belong to dense regions receive large score values. In the second step, based on the estimated scores, we develop a statistical method that exploits the beta mixture model to automatically discriminate outliers from nodes that lie within dense regions. The two steps are described in detail below.

2.2.1 Step 1: Estimating Outlier Scores

In this subsection, we devise a method to estimate an outlier score for each node in order to capture the difference between normal nodes (that is, nodes that lie within communities) and outlier nodes. To this end, we first develop a function that reflects the strength of connections between nodes in the multidimensional setting. The proposed function is based on the assumption that normal nodes are strongly connected across different subspaces of dimensions, whereas outliers are weakly connected to the rest of the network's nodes. Then, we evaluate the strength of connection of each node with respect to its immediate neighboring nodes in order to estimate an outlier score that helps to distinguish between normal and outlier nodes.

Generally, normal nodes tend to form dense regions along different combinations of dimensions, while outlier nodes are sparsely distributed across all the network dimensions. Accordingly, nodes within dense regions share a common pattern of interaction in the sense that they may very likely have a relatively large number of common neighbors, which is not the case for outlier nodes. With this intuition in mind, we define the strength of connection between two nodes u and v as

$$f(v, u) = \frac{\mid \eta(u, D(v, u)) \cap \eta(v, D(v, u)) \mid}{\mid \eta(u) \cap \eta(v) \mid} \tag{1}$$

where $D(v, u) \subseteq D$ is the subset of dimensions that only connect nodes u and v. In Eq. (1), $\eta(u, D(v, u))$ denotes the set of neighboring nodes of u with respect to the dimensions in $D(v, u)$, while $\eta(u)$ represents the set of neighbors of u with respect to all the network dimensions D.

To illustrate, consider again the three-dimensional network depicted in Fig. 2. This network contains nine nodes connected along three dimensions ($D = \{d_1, d_2, d_3\}$). Let's examine, for example, the neighbors of nodes n_4 and n_5 which are connected through dimensions d_1 and d_2. For this case, $D(n_4, n_5) = \{d_1, d_2\}$, $\eta(n_4, D(n_4, n_5)) = \{n_5, n_6, n_7, n_8\}$ and $\eta(n_5, D(n_4, n_5)) = \{n_4, n_6, n_7\}$. On the other hand, the neighbors of node n_4 across all the network dimensions are $\eta(n_4) = \{n_5, n_6, n_7, n_8\}$, while neighbors of node n_5 in D are $\eta(n_5) = \{n_4, n_6, n_7, n_8\}$. In the following, we explain the reasoning behind the definition of $f(u, v)$ as described by Eq. (1).

Given that u and v are connected, the term $\mid \eta(u, D(u, v)) \cap \eta(v, D(u, v)) \mid$ corresponds to the number of shared neighbors between u and v along the set of dimensions in $D(u, v)$, while the term $\mid \eta(u) \cap \eta(v) \mid$ computes the number of shared neighbors between nodes u and v across all the dimensions in D. The strength of connection between u and v, as estimated by $f(u, v)$, corresponds to the ratio of the number of neighbor nodes common between u and v along the set of dimensions that exclusively connects u and v (that is, $D(u, v)$) to the number of shared nearest neighbors between u and v across all the network dimensions (that is, D). By doing so, $f(u, v)$ evaluates the connectivity between nodes u and v based on the dimensions they use and their topological structure regarding their neighborhoods. In such a way, the function $f(u, v)$ reveals the connection's strength between pairs of nodes as it describes how they are tightly connected and how similar are their neighborhoods.

The values of $f(u, v)$ are always between 0 and 1. The highest value of $f(u, v)$ suggests that nodes u and v share a relatively large number of common neighbors along the subset of dimensions $D(u, v) \subseteq D$. On the other hand, a small value of $f(u, v)$ indicates that u and v are sparsely connected in the sense that they share a very few number of common neighbors with respect to $D(u, v)$. A null value of $f(u, v)$ indicates that u and v do not have any neighboring nodes in common. $f(u, v)$ provides, thus, a relative measure to relate the connection's strength between nodes so that it would be easily possible to spot outlier nodes with an irregular pattern of interactions and random connections. For the purpose of illustration, consider nodes n_0 and n_8 in the network depicted in Fig. 2. The strength of connection of these nodes to their neighbors as estimated by the function f described by Eq. (1) is 0. That is, $f(n_0, n_1) = f(n_0, n_6) = 0$ and

$f(n_8, n_4) = f(n_8, n_5) = 0$. On the other hand, any pair of the remaining nodes (that is, $n_1, n_2, n_3, n_4, n_5, n_6$, and n_7) that share common neighbors with respect to their connecting dimensions are characterized by high $f(u, v)$ values (mostly equal to 1). Here, we can claim that the $f(u, v)$ values help to fairly well discriminate between closely connected nodes and loosely connected ones.

Outlier nodes tend to be weakly connected to the remaining nodes of the network across all the dimensions. In other words, all the values of $f(u, v)$ associated with an outlier tend to very small values (close to 0). In this setting, for a specific node u, calculating the average value of $f(u, v)$ with respect to its immediate neighbors across D is a fair indicator of an outlierness degree of u. Specifically, an outlier score $OS(u)$ of a node u can be estimated as

$$OS(u) = \frac{\sum\limits_{v \in \eta(u)} f(u, v)}{\mid \eta(u) \mid} \tag{2}$$

As described by Eq. (2), $OS(u)$ computes the average connection strength of the neighbors of node u in D. Accordingly, nodes that share a large number of common neighbors will get a large value of $OS(u)$ in comparison to sparsely connected nodes that do not have any (or a very low number of) common neighbors with respect to the rest of the network nodes. As a result, for a potential outlier node, $OS(u)$ will be the smallest, in comparison to densely connected nodes. For example, nodes n_0 and n_8 in the network depicted in Fig. 2 have the lowest anomaly score: $OS(n_0) = OS(n_8) = 0$, whereas the remaining nodes (which are closely connected) have relatively high score values (close to 1). $OS(u)$ provides, thus, a relative measure of the outlier score, which in turn facilitates the discrimination between outliers and normal nodes.

2.2.2 Step 2: Automatic Identification of Outliers

Let us focus now on how to automatically identify outlier nodes based on $OS(u)$. As just discussed, the values of $OS(u)$ differ in such a way that outliers are characterized by relatively low scores in comparison to nodes that belong to communities. Finite mixture models are typically used to analyze data of this type [26, 27]. Specifically, the estimated $OS(u)$ can be considered as coming from several underlying probability distributions. Each distribution is a component of the mixture model representing data points with close $OS(u)$ values, and all the components are combined into a comprehensive model by a mixture form. In this paper, we propose to use the beta mixture model to divide the outlier scores into a number of components so that the smallest scores can be identified.

We use the beta mixture model because it permits multiple modes and asymmetry and can thus approximate a wide variety of shapes [27–29] while several other distributions are not able to do so. For example, the Gaussian distribution permits a symmetric bell shape only. However, in many applications, the data under investigation is skewed with non-symmetric shapes. As observed in [30], due to its symmetric shape restriction, the standard Gaussian distribution may lead to inaccurate modeling (e.g., over estimation of the number of components in the mixture, increase of misclassification errors, etc.). In contrast to several distributions, the beta distribution is more flexible and powerful since it permits multiple symmetric and asymmetric modes, and it may be skewed to the right, skewed to left, or symmetric [27–29]. The shape of the Gaussian, and other distributions such as the Gamma and uniform distributions, is thus a special case of the beta distribution. This great shape flexibility of the beta distribution provides a better fitting of the anomaly scores, which leads, in turn, to accurate detection of anomalous nodes.

Since the beta distribution is defined in the compact support [0,1], $OS(u)$ values are thus normalized between 0 and 1 without changing the main statistical properties. Let ω_i, $(i = 1, \ldots, n)$, denote the normalized scores (recall that n is the total number of nodes in the network under investigation). Formally, we expect that $\{\omega_i\}$ follow a mixture density of the form

$$F(\omega) = \sum_{l=1}^{p} \alpha_l B_l(\omega, x_l, y_l), \tag{3}$$

where $B_l(.)$ is the lth beta distribution, p denotes the number of components in the mixture, x_l and y_l $(x_l, y_l > 0)$ are the shape parameters of the lth component, α_l $(l = 1, \ldots, p)$ are the mixing coefficients, with the restriction that $\alpha_l > 0$ for $l = 1, \ldots, p$ and $\sum_{l=1}^{p} \alpha_l = 1$. The density function of the lth component is given by

$$B_l(\omega, x_l, y_l) = \Psi(x_l, y_l)\omega^{x_l-1}(1-\omega)^{y_l-1} \tag{4}$$

where $\Psi(x_l, y_l) = \frac{\Gamma(x_l+y_l)}{\Gamma(x_l)\Gamma(y_l)}$ and $\Gamma(.)$ is the gamma function given by $\Gamma(\lambda) = \int_0^\infty t^{\lambda-1}\exp(-t)dt;\ t > 0$.

A common approach for estimating the parameters x_l and y_l of the beta component is the maximum likelihood technique [29]. The likelihood function of the lth component is defined as

$$L_{B_l}(x_l, y_l) = \prod_{\omega \in B_l} B_l(\omega, x_l, y_l) = \left(\frac{\Gamma(x_l+y_l)}{\Gamma(x_l)\Gamma(y_l)}\right)^{n_l} \prod_{i=1}^{n_l}(\omega_i)^{x_l-1} \prod_{i=1}^{n_l}(1-\omega_i)^{y_l-1} \tag{5}$$

where n_l is the size of the lth component. The logarithm of the likelihood function is given by

$$\log(L_{B_l}(x_l, y_l)) = n_l \log(\Gamma(x_l + y_l)) - n_l \log(\Gamma(x_l)) - n_l \log(\Gamma(y_l)) + (x_l - 1) \sum_{i=1}^{n_l} \log(\omega_i) + (y_l - 1) \sum_{i=1}^{n_l} \log(1 - \omega_i) \tag{6}$$

To find the values of x_l and y_l that maximize the likelihood function, we differentiate $\log(L_{B_l}(x_l, y_l))$ with respect to each of these two parameters and set the result equal to zero:

$$\frac{\partial}{\partial x_l} \log(L_{B_l}(x_l, y_l)) = \frac{n_l \Gamma'(x_l + y_l)}{\Gamma(x_l + y_l)} - \frac{n_l \Gamma'(x_l)}{\Gamma(x_l)} + \sum_{i=1}^{n_l} \log(\omega_i) = 0 \tag{7}$$

and

$$\frac{\partial}{\partial y_l} \log(L_{B_l}(x_l, y_l)) = \frac{n_l \Gamma'(x_l + y_l)}{\Gamma(x_l + y_l)} - \frac{n_l \Gamma'(y_l)}{\Gamma(\beta_l)} + \sum_{i=1}^{n_l} \log(1 - \omega_i) = 0 \tag{8}$$

The parameters $\hat{x_l}$ and $\hat{y_l}$ can be estimated by solving the system of Eqs. (7) and (8) using the Newton–Raphson method.

The use of the beta distribution mixture yields a flexible model to describe the distribution of the outlier scores. To form such a model, we need to estimate p, the number of components, and the parameters for each component. One popular approach to specifying the number of components p is to increase p from 1 to p_max (the maximal number of components in the mixture) and to compute certain performance measures in each run, until a partition into an optimal number of components is obtained. For this purpose, we implement a standard two-step process. In the first step, we calculate the maximum likelihood of the parameters of the mixture for a range of values of p (from 1 to p_max). The second step involves calculating the associated criterion and selecting the value of p which optimizes the criterion. A variety of approaches have been proposed to estimate the number of components in the data [31]. In our method, we use the penalized likelihood criterion, called the Bayesian information criterion (BIC). BIC was first introduced by Schwarz [32] and is given by

$$BIC(p) = -2L_p + Nb_p \log(N) \tag{9}$$

where L_p is the logarithm of the likelihood at the maximum likelihood solution of the model under investigation, and Nb_p is the number of parameters estimated. The number of components that minimizes $BIC(p)$ is considered to be the optimal value of p.

Typically, the maximum likelihood of the parameters of the distribution is estimated using the expectation-maximization (EM) algorithm. This algorithm

Algorithm 1: Estimation of p

```
Input  : $\{\omega_i\}$, $p\_max$
Output: The optimal number of components $p$
begin
    for $p = 1$ to $p\_max$ do
        if $p==1$ then
            Estimate $\hat{x}$ and $\hat{y}$ using the Newton–Raphson method based on (7) and (8);
            Compute the value of $BIC(p)$ using (9);
        else
            Apply FCM as an initialization of the EM algorithm;
            Apply the EM to estimate the parameters of the mixture $\hat{x}_l$ and $\hat{y}_l$
                $(l = 1, \ldots, p)$;
            Compute the value of $BIC(p)$ using (9);
        end
    end
    Select the number of components $\hat{p}$, such that $\hat{p} = \arg_{min} BIC(p)$;
end
```

requires the initial parameters of each component. Since EM is highly dependent on initialization, it will be helpful to perform initialization by means of a clustering algorithm [33]. For this purpose we implement the fuzzy C-means (FCM) algorithm [34] to partition the set $\{\omega_i\}_{i=1,\ldots,n}$ into p components. Based on this partition we can estimate the parameters of each component and set them as initial parameters to the EM algorithm. The procedure for estimating the number of components is summarized in Algorithm 1.

Let's now discuss the choice of the value of p_max. Based on extensive experiments on various networks, we found that, in most cases, the optimal number of components in a mixture varies from 2 to 3. This result can be explained by the fact that the outlier score provides a relative measure on which outlier nodes are easily distinguishable from nodes that belong to dense structures. Such a concept of relativity between the values of the outlier scores makes the choice of p_max fairly simple. Based on this, we believe that setting $p\_max = 5$ is, in general, a practical choice. However, the reader should be aware that the value of p_max is not limited to 5 and the user can set any other value. In our experiments, we fixed $p\_max = 5$.

2.3 Summary of Outlier Handling Procedure

Once the optimal number of components is identified, we can use the results of the EM algorithm to derive a classification decision about the membership of ω_i in each component in the mixture. To identify outlier nodes, we are interested in the beta component that corresponds to the smallest values of ω_i. Accordingly, nodes associated with the set of values of ω_i that belong to such a component correspond to outliers. The identified outlier nodes are then discarded from the original network G and stored in the set OUT. Thus, phase 1 of MCDA yields a reduced network RG

Algorithm 2: Phase 1 of MCDA

```
Input  : $G$ : the original multidimensional network
Output: $OUT$ : the set of outlier nodes
        $RG$ : the multidimensional network without outliers
begin
    For each node $u$ in $G$ estimate $OS(u)$ using (2);
    Estimate $\{\omega_i\}_{i=1,\dots,n}$ by normalizing the values $OS(u)$ between 0 and 1;
    Using Algorithm 1, estimate the probability density function of the normalized anomaly
        scores with different values of $p$ where $p = 1, \dots, p\_max$;
    Select the mixture model with the optimal number of components that minimize $BIC$;
    Use the results of EM to derive a classification decision about the membership of $\omega_i$ in
        each component;
    Select the beta component that corresponds to small values of $\omega_i$;
    Identify nodes in $G$ associated with $\omega_i$ that belong to the selected component and store
        them in $OUT$;
    Extract $RG$ such that $RG \leftarrow G - OUT$;
    Return $OUT$ and $RG$;
end
```

with size $z = n - |OUT|$. Note that, unless otherwise specified, in the remainder of this paper, we will focus on detecting community structures from the network RG that does not contain outliers. Our method for eliminating outliers is described in Algorithm 2.

For the purpose of illustration, consider the synthetic network presented in Fig. 1. Recall that this dataset contains 400 nodes that lie within communities and 50 outlier nodes. For each node u in this network, we have first calculated $OS(u)$ using Eq. (2). Next, we estimated $\{\omega_i\}$ by normalizing the values of $OS(u)$ between 0 and 1. Finally, based on Algorithms 1 and 2, we estimated the probability density function of $\{\omega_i\}$ and identified the beta component that corresponds to outliers. Figure 3 depicts the estimated density curves. As we can observe from this pictorial illustration, the beta distribution has a great shape flexibility which allows, in turn, accurate modeling of the outlier scores. The first component in the plot depicted in Fig. 3 represents the lowest score values. Nodes associated with the scores grouped in this component correspond to outlier nodes.

2.4 Mining Community Structures

The second phase of MCDA handles the recovery of the hidden community structures from the multidimensional network RG. Here, the problem is twofold: we must discover the communities and find the appropriate set of dimensions in which each community exists. To tackle this "chicken-and-egg" problem, we proceed in two steps. In the first step, we cluster the network nodes by devising a parameterless local search algorithm. Then, based on the communities obtained in the first step,

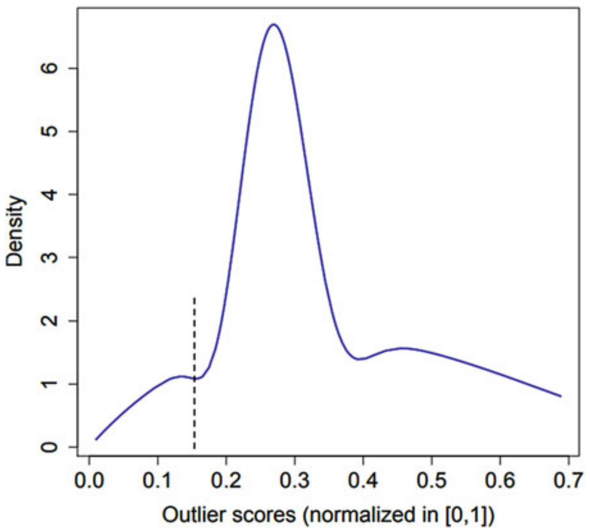


Fig. 3 Density curve of the outlier score corresponding to the synthetic network depicted in Fig. 1. The first component (the one close to zero, separated by the dashed line) corresponds to outliers component

the second step proceeds to select relevant dimensions of the identified communities. Below, we describe our community detection approach and then devise a method for the automatic selection of relevant dimensions for each community.

2.4.1 Community Detection Approach

The community detection approach we propose in this paper is inspired from the label propagation principle. In essence, a label propagation algorithm (LPA) [35–38] is a fast local search technique that relies on the network structure as a lone guide for its community search process. The LPA's procedure starts from an initial state where each node is assigned a unique numerical label that represents its community membership. Nodes are then visited to update their memberships based on the dominant community labels in their neighborhoods. The relabeling process keeps repeating until all nodes are assigned the community label that dominates their neighborhoods. Communities are then extracted from the node sets bearing the same label. In general, each LPA variant considers the dominance of a given membership label according to the propagation rule it adopts. For instance, the basic LPA [35] selects the most frequent label from among the node's neighbors. For a simple monodimensional network G', the propagation rule of the basic LPA can formally be expressed as:

$$l'_v = \arg\max_{\ell} \sum_{u \in \eta(v)} A_{vu}\ \delta(l_u, \ell) \tag{10}$$

where l'_v denotes the new community membership label of the processed node $v \in V'$; A_{vu} an element of the binary adjacency matrix of G'; $\eta(v)$ the set of v's neighbors; l_u is the community membership label of node u; and δ is the Kronecker delta, a function that returns 1 if ℓ and l_u are equal. Otherwise, the function returns zero. In case two or more values equally maximize the sum, arg max must keep the current label l_v of v if it is already among the dominant labels or otherwise take a random one from the dominant group. The propagation rule described by (10) updates the membership labels so as to increase the number of edges inside a community. This definition is based on the intuition that nodes forming a community tend to have more neighbors internally than elsewhere.

In order to support the multidimensional setting, it is important to take into consideration the fact that each dimension might contribute differently to the formation of the communities. In fact, defining the dominance of a membership label in a neighborhood using the same intuition used in monodimensional networks, that is, using the number of contributed edges to the neighboring community, as performed by LPA on the aggregated representation [12], could lead to a poor performance. This is particularly true in the case of networks in which the number of irrelevant dimensions is higher. Therefore, any propagation rule that operates on a multidimensional network must consider the relevance of dimensions to which the edges being counted belong. This raises the question of how to define the relevance of a dimension to the neighboring community and the processed node.

Intuitively, nodes belonging to a multidimensional community are expected to interact sufficiently within its subspace of relevant dimensions. Therefore, the density of edges belonging to the relevant dimensions is expected to be higher than it is with irrelevant dimensions. Nodes forming a multidimensional community would thus predominantly use a common subset of relevant dimensions to reach out to each other. These subsets can be recovered at the node level by locating the most frequently used dimensions for interaction with the neighboring nodes. One can exploit this information as an additional constraint for membership selection in such a way that neighbors which connect with the visited node through the maximum number of relevant edges define its new membership. This constraint can be achieved through a weighting strategy which assigns to each neighbor an affinity score based on the number of relevant dimensions connecting the pair. Such a rule can be expressed as:

$$l'_v = \arg\max_{l} \sum_{u \in \eta(v)} as(v, u)\ \delta(l_u, l) \tag{11}$$

where $as(v, u)$ denotes the affinity score of the visited node v to its neighbor u. A higher value of $as(v, u)$ indicates that v is connected to u through a higher number of relevant dimensions for v and u simultaneously, whereas a low value indicates a low number of relevant dimensions for either of the nodes. Here, it is important to stress that the number and the relevance of dimensions connecting the pair must be considered for both v and u. Specifically, based on the dimensions connecting (v, u), $as(v, u)$ must be estimated so as to satisfy the following two constraints: (1) the higher the number of relevant dimensions to v, the larger is the value of $as(v, u)$, and (2) the higher the number of relevant dimensions to its neighbor u, the larger is the value of $as(v, u)$. These two constraints must be honored by $as(v, u)$ in order to successfully discriminate between adjacent nodes belonging to different communities. The next sections present the strategy that we have developed for the estimation of the as scores. The proposed strategy involves a two-stage process in which each constraint is expressed separately. First, $as(v, u)$ is measured by considering the number of relevant dimensions to the node v. Next, an adjustment is carried out to reflect the relevance of the connecting dimensions to the neighbor u.

Initial Estimation of the Affinity Scores In order to estimate the affinity score $as(v, u)$ for a node v to its neighbor u, we initially rely on the number of relevant dimensions connecting the pair from v's perspective. To this end, we adopt the dimension relevance xOR metric (DR_{xOR}) [3], a function that evaluates the relevance of a group of dimensions $S \subseteq D$ for a node v based on the ratio of v's neighbors that can be exclusively reached in any subset of S. Formally, the DR_{xOR} is defined as:

$$DR_{xOR}(v, S) = \frac{|\eta_{xOR}(v, S)|}{|\eta(v)|} \tag{12}$$

Recall that $\eta(v)$ is the set of v's neighbors with respect to all the network dimensions D and $\eta_{xOR}(v, S)$ is the set of v's neighbors exclusively reachable in any subset $S_i \subseteq S$ and is defined as $\eta_{xOR}(v, S) = \{u | \exists(v, u, s) \in E \wedge s \in S \wedge \forall d \in D - S, \nexists(v, u, d) \in E\}$. The DR_{xOR} returns values in [0, 1] and achieves its maximum when all v's neighbors cannot be reached outside S. This metric is suitable for the estimation of the affinity scores since it favors larger groups of relevant dimensions to v. In what follows, we define $ias(v, u)$, the initial affinity score of node v to neighbor u by:

$$ias(v, u) = DR_{xOR}(v, D(v, u)) \tag{13}$$

Recall that $D(v, u)$ denotes the set of dimensions that only connect v and u. A higher value of $ias(v, u)$ indicates a higher number of relevant dimensions within $D(v, u)$ to v and, initially, a higher chance of v being in u's community. However, since it does not honor the second constraint, a high value of the estimated ias does not necessarily guarantee that v would contribute a higher number of relevant edges to the formation of u's community. Therefore, the relevance of the connecting dimensions to the neighbors of v must be expressed in the affinity scores. In the following, we introduce a revision procedure for $ias(v, u)$ to enforce the second constraint.

Revision of the Initial Affinity Scores The second step in the estimation of $as(v, u)$ deals with the expression of the relevance of $D(v, u)$ to the neighbor u. A possible way to achieve this is to penalize the initially estimated value ias by the distance separating $D(v, u)$ from the relevant dimensions to the neighbor u. To this end, we define $D_u \subseteq D$, the set of relevant dimensions to the neighbor u. Recall that a high value of ias reflects a high number of relevant dimensions to v. Hence, one possible way to recover the relevant dimensions D_v for each node $v \in V'$, where $V' = V - OUT$, is to select all $D(v, u)$ for which the combined score ias is the highest. Given a node $v \in V'$, we define the set of relevant dimensions D_v as:

$$D_v = \arg\max_{S} \sum_{u \in \eta(v)} ias(v, u)\ \delta(D(v, u), S) \tag{14}$$

In case two or more sets equally maximize the sum, arg max should take their union. Now that D_u is defined, we can estimate the revised affinity score $as(v, u)$ of v to u using the distance between $D(v, u)$ and D_u. To this end we rely on the Jaccard coefficient. Formally, $as(v, u)$ is defined as:

$$as(v, u) = ias(v, u) \times \frac{|D_u \cap D(v, u)|}{|D_u \cup D(v, u)|} \tag{15}$$

The higher the value of $as(v, u)$, the higher is the number of relevant dimensions within $D(v, u)$ for v and u jointly. Algorithm 3 summarizes the steps for the estimation of the affinity scores as.

Adopting the weighting strategy in Eq. (15) and the propagation rule in Eq. (11), a label propagation-based process can be used to support the discovery of multidimensional communities. Specifically, the process starts by assigning a unique community label l_v to each node v. Nodes are then asynchronously processed according to Eq. (11). Each propagation step involves the estimated scores as in the selection of the new membership of the processed node. This iterative process keeps running until the stop criterion is satisfied, that is, when all nodes are assigned the community labels that support the highest affinity scores in their neighborhoods

Algorithm 3: Affinity scores estimation

```
Input  : $RG$
Output: affinity scores matrix $as$
begin
    // Initial estimation of the affinity scores
    foreach $v \in V'$ do
        foreach $u \in \eta(v)$ do
            Calculate $ias(v, u)$ according to (13);
        end
        Select $D_v$ according to (14);
    end
    // Revision of the initial affinity scores
    foreach $v \in V'$ do
        foreach $u \in \eta(v)$ do
            Calculate $as(v, u)$ according to (15);
        end
    end
    Return $as$;
end
```

Algorithm 4: Communities detection procedure

```
Input  : $RG$
Output: $\{C_k\}_{k=1...K}$
begin
    Estimate $as$ according to Algorithm 3;
    foreach $v \in V'$ do
        Assign a unique community label $l_v$ for $v$;
    end
    // Identification of $l_v$ for each node $v \in V'$
    while $\exists v \in V'$ such that $l_v$ is different from the dominant label in $\eta(v)$ do
        foreach $v \in V'$ do
            Update $l_v$ according to Eq. (11);
        end
    end
    // Identification of communities
    Group nodes $v \in V'$ with similar label $l_v$ into communities $\{V_k\}_{k=1...K}$, ($K$ is the
      number of the identified groups);
    Return $\{C_k = (V_k)\}_{k=1...K}$ ;
end
```

according to Eq. (11). In such a case, the resulting communities can be recovered from the nodes bearing the same community label. Algorithm 4 summarizes the proposed communities detection procedure.

Finally, we should point out that our approach is nondeterministic due to its parallel processing of the nodes list and the random memberships selection when more than a single membership label is equally dominating the neighborhood of a visited (processed) node. However, as the knowledgeable reader will observe from the experimental results, the impact that poses this nondeterministic aspect to MCDA is not significant.

2.4.2 Selection of Relevant Dimensions of Communities

Once community structures are discovered, MCDA deals with the explicit selection of their relevant dimensions. The idea is to define, for each dimension $d \in D$, a relevance index $R(d, C_k)$ that determines how well it contributes to the formation of a community C_k formed by nodes in $\{V_k\}$. Based on the estimated relevance scores, we propose a method to automatically discriminate relevant dimensions of a community from irrelevant ones. As discussed earlier, nodes belonging to the same community are densely connected across their relevant dimensions. In other words, nodes in C_k must exhibit a higher internal density of links along relevant dimensions. In this setting, the index $R(d, C_k)$ for the dimension d in community C_k can be defined as:

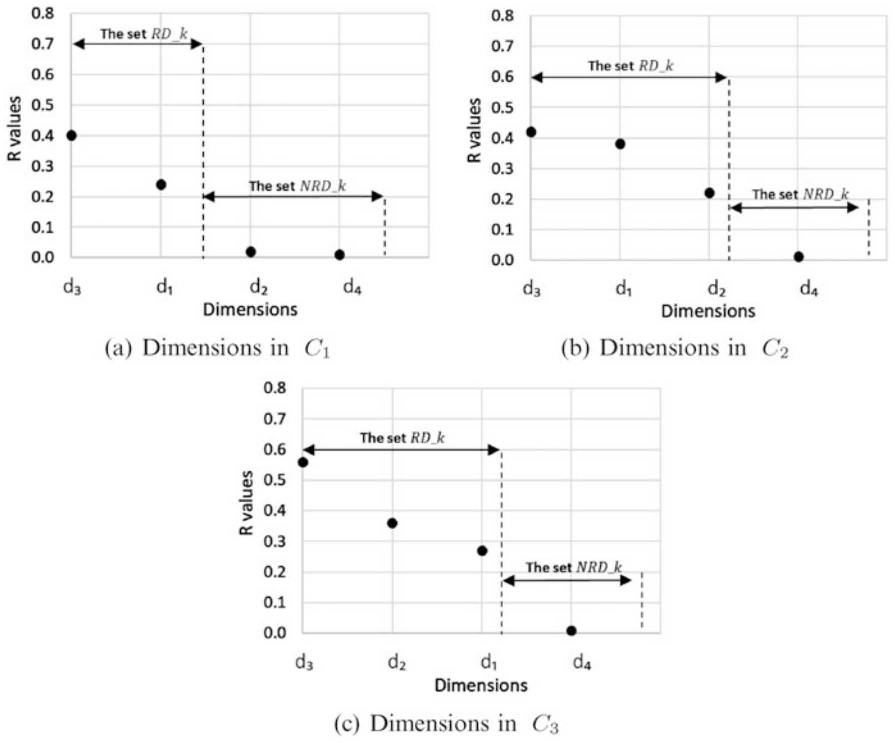
$$R(d, C_k) = \frac{\sum_{v,u \in V_k} A^d_{vu}}{|C_k| \times (|C_k| - 1)} \tag{16}$$

where A^d_{vu} denotes an element of the binary adjacency matrix of the graph projected along dimension d. The relevance index, as defined by Eq. (16), is based on the formula of the internal density of links belonging to dimension d that connect the nodes in V_k. A high value of $R(d, C_k)$ suggests that nodes in V_k are densely connected within dimension d. On the other hand, a small value of $R(d, C_k)$ suggests that nodes in V_k are sparsely connected along d. Note that in the case of a singleton community, that is, a community with one node only, the relevant dimensions are not defined.

Once the relevance index $R(d, C_k)$ of each dimension is estimated, we turn our attention to the problem of selecting the subset of relevant dimensions $D_k \subseteq D$ supporting the formation of C_k. As just discussed, relevant dimensions $d \in D_k$ of a community are characterized by large values of $R(d, C_k)$ while irrelevant ones are characterized by low values. In order to identify the set D_k supporting the formation of C_k, we are interested in all dimensions $d \in D$ with large values of $R(d, C_k)$. To this end, we rely on an inter-class inertia-based procedure to automatically separate the highest relevance score values from the lowest ones. Specifically, let $R_{C_k} = \{R(d, C_k)\}_{d \in D}$ be the set of all relevance scores of all dimensions $d \in D$ for a community C_k. Our goal is to divide R_{C_k} into two groups RD_k and NRD_k where RD_k contains the highest values of $R(d, C_k)$, and NRD_k the lowest values. This is achieved by computing the partition into two sets which have the maximum inter-class inertia on a subset of all possible partitions. To this end, we implement the iterative model described in Algorithm 5. Based on the obtained results, we select the relevant dimensions D_k associated with the values in RD_k.

Algorithm 5: Selection of relevant dimensions D_k

```
Input  : $R_{C_k}$
Output: $D_k$
begin
    Sort the relevance scores in $R_{C_k}$ in a descending order;
    Compute $\mu_{R_{C_k}}$, the mean of $R_{C_k}$;
    for i=1 to o do
        Split $R_{C_k}$ into two sets $RD_k$ and $NRD_k$ containing the first $i$ and last $(o - i)$
            relevance score values respectively;
        Compute $\mu_{RD_k}$, the mean of $RD_k$;
        Compute $\mu_{NRD_k}$, the mean of $NRD_k$;
        Compute the inter-class inertia
            $I(i) = |RD_k| \times (\mu_{R_{C_k}} - \mu_{RD_k})^2 + |NRD_k| \times (\mu_{R_{C_k}} - \mu_{NRD_k})^2$;
    end
    Select the best partitioning given by the pair $(RD_k, NRD_k)$, that is, the one for which
        the corresponding $I(i)$ is the highest;
    Select the dimensions corresponding to each of $R(d, C_k)$ values in $RD_k$ and store them
        in $D_k$;
    Return $D_k$;
end
```



(a) Dimensions in C_1 (b) Dimensions in C_2 (c) Dimensions in C_3

Fig. 4 The relevance scores for the dimensions of the four-dimensional network depicted in Fig. 1 with respect to (**a**) C_1 and (**b**) C_2. The relevant dimensions of C_1 are d_1 and d_3 while the relevant dimensions of C_2 and C_3 are d_1, d_2, and d_3. (**a**) Dimensions in C_1. (**b**) Dimensions in C_2. (**c**) Dimensions in C_3

Figure 4 illustrates the relevance scores of the four dimensions (sorted in descending order) of the three communities recovered from the four-dimensional network depicted in Fig. 1. As shown in Fig. 4, there is a clear cutoff point which allows us to select, for each community, the dimensions with the highest relevance score values.

2.5 *Summary of the MCDA Approach*

Algorithm 6 summarizes the proposed approach. The steps described in this algorithm can be implemented to effectively identify the set OUT of outliers nodes, the set V_k of nodes that form each community C_k, and the set D_k of its relevant dimensions.

Algorithm 6: The MCDA approach

```
Input  : The multidimensional network $G$
Output: Communities and their relevant dimensions $\{C_k = (V_k, D_k)\}_{k=1...K}$
begin
    // Phase 1: Detect and eliminate outliers
    Apply Algorithm 2 $(G, OUT, RG)$ to detect $OUT$ and extract $RG$ from $G$;
    // Phase 2: Detect communities and their relevant
       dimensions
    // Identify communities in $RG$
    Apply Algorithm 4 $(RG, C_k)$ to detect the set of communities $\{C_k = (V_k, D_k)\}_{k=1...K}$
    in $RG$;
    // Identify the set of relevant dimensions $D_k$ of each
       community $C_k$
    foreach $C_k$ do
        Estimate the set of relevance scores $R_{C_k} = \{R(d, C_k)\}_{d \in D}$ using (16);
        Apply Algorithm 5 $(R_{C_k}, D_k)$ to identify $D_k$;
    end
    // Return communities and their selected dimensions
    Return $\{C_k = (V_k, D_k)\}_{k=1...K}$ ;
end
```

3 Experimental Results

This section reports the performance results of MCDA on various synthetic and real networks. To this end, four approaches were selected for comparison: PMM [13], SC-ML [18], GraphFuse [10], and ensemble clustering [14]. For the latter, the communities are first recovered from each dimension separately using the Louvain method [21]. A consensus partition is then constructed from the obtained partitions on the individual dimensions. The consensus partition is selected based on the best results of the three consensus methods proposed in [14] namely, CSPA (cluster-based similarity partitioning algorithm), HGPA (hyper graph partition algorithm), and MCLA (meta clustering algorithm). It is important to note that other approaches such as ABACUS [15] and LART [25] could not be considered in the comparison since their outputs differ significantly from the output of the selected comparing algorithms. In fact, as discussed in Sect. 1.1, approaches in [15, 25] return a large number of densely overlapping communities across multiple subspaces. This makes the comparison not obvious with MCDA, PMM, SC-ML, and ensemble clustering since these later approaches identify disjoint multidimensional community structures.

3.1 Experiments on Synthetic Networks

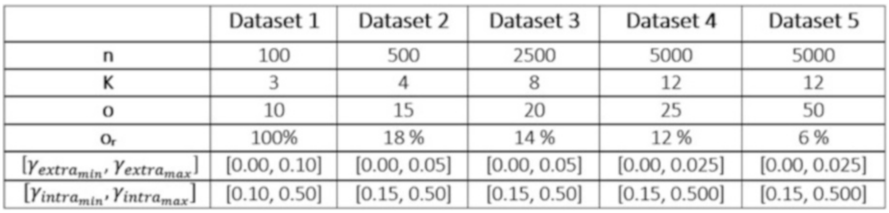
The goal of the experiments conducted in this section is to evaluate the suitability of MCDA in terms of: (1) accuracy—the aim is to test whether our algorithm, in comparison with other existing approaches, is able to correctly identify multidimensional communities, and (2) efficiency—the aim is to determine how the running time scales with the size and the dimensionality of the network. For this purpose, we generated a variety of synthetic datasets to simulate various situations, using the data generation model described below.

3.1.1 Synthetic Network Generation

Synthetic networks were artificially generated according to a procedure that is based on the planted partitions model [39]. Specifically, the generation process was made parametric to the number of nodes n, the number of communities K, the number of dimensions o, the average community dimensionality o_r, the range of the intra-community densities of links $[\gamma_{intra\_min}, \gamma_{intra\_max}]$, the range of the inter-community densities of links $[\gamma_{extra\_min}, \gamma_{extra\_max}]$, and the percentage of outliers PO. Based on the provided parameters, the communities are planted across the network's dimensions in such a way that each community exhibits various densities of links across its relevant dimensions. Specifically, a subset of relevant dimensions D_r is selected such that $|D_r| \geq o_r$. For each $d \in D_r$, the corresponding adjacency matrix is split into K blocks B_k^d of different sizes. Each block B_k keeps the same size regardless of $d \in D_r$. For each community C_k, a subset of blocks B_k^d is randomly selected across D_r such that the average number of relevant dimensions for the planted communities remains close to o_r. Nodes within each selected block are then wired randomly according to a probability uniformly sampled from $[\gamma_{intra\_min}, \gamma_{intra\_max}]$. Inter-community edges are then added according to another probability, which is uniformly drawn from $[\gamma_{extra\_min}, \gamma_{extra\_max}]$. The adjacency matrices of the remaining irrelevant dimensions, that is $D - D_r$, are constructed following the Erdös–Rényi model with an edge generation probability uniformly sampled from $[\gamma_{extra\_min}, \gamma_{extra\_max}]$ for each dimension. Finally, note that outliers are injected in such a way that they are sparsely distributed across all the network dimensions and the density of their connections conforms to the background noise density (in irrelevant dimensions), which is uniformly drawn from $[\gamma_{extra\_min}, \gamma_{extra\_max}]$.

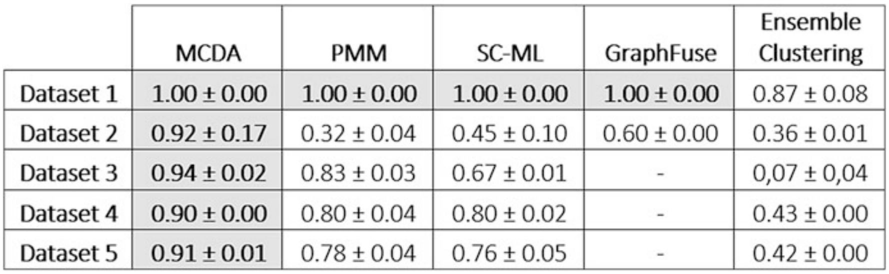
3.1.2 Community Detection Accuracy

The main concern of this first set of experiments was to compare the detection accuracy of MCDA to that of PMM, SC-ML, GraphFuse, and ensemble clustering. To this end, we generated five different synthetic networks. Figure 5 summarizes the used parameters for the generation of the synthetic networks of our experiments. Each network exhibits a different configuration with respect to the number of dimensions, nodes, and communities, as well as their relevant dimensions. As we can see from Fig. 5, the number of nodes n varies from 100 to 5000, while the average community dimensionality o_r varies from 6 to 100% of the whole dimensionality o. The dimensions of Dataset 1 are all considered relevant for its



	Dataset 1	Dataset 2	Dataset 3	Dataset 4	Dataset 5
n	100	500	2500	5000	5000
K	3	4	8	12	12
o	10	15	20	25	50
o_r	100%	18 %	14 %	12 %	6 %
$[\gamma_{extra_{min}}, \gamma_{extra_{max}}]$	[0.00, 0.10]	[0.00, 0.05]	[0.00, 0.05]	[0.00, 0.025]	[0.00, 0.025]
$[\gamma_{intra_{min}}, \gamma_{intra_{max}}]$	[0.10, 0.50]	[0.15, 0.50]	[0.15, 0.50]	[0.15, 0.500]	[0.15, 0.500]

Fig. 5 Generation parameters for the synthetic networks



	MCDA	PMM	SC-ML	GraphFuse	Ensemble Clustering
Dataset 1	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	0.87 ± 0.08
Dataset 2	0.92 ± 0.17	0.32 ± 0.04	0.45 ± 0.10	0.60 ± 0.00	0.36 ± 0.01
Dataset 3	0.94 ± 0.02	0.83 ± 0.03	0.67 ± 0.01	-	0,07 ± 0,04
Dataset 4	0.90 ± 0.00	0.80 ± 0.04	0.80 ± 0.02	-	0.43 ± 0.00
Dataset 5	0.91 ± 0.01	0.78 ± 0.04	0.76 ± 0.05	-	0.42 ± 0.00

Fig. 6 Performance of compared algorithms on synthetic networks. Shaded regions correspond to the best results

three communities. On the other hand, on average, only 6% of dimensions are relevant to the communities of Dataset 5. The generated networks allow us to study the impact of community dimensionality on the quality of results. Note that no outliers were generated since our goal in this first set of experiments was to evaluate the robustness of the competing algorithms in situations that involve the presence of a high number of irrelevant dimensions. In addition to this, as discussed in Sect. 1, existing approaches, such as the ones considered in our experiments, do not have an outlier detection mechanism. The outlier detection mechanism of MCDA was therefore disabled.

The adopted generation model allows us to produce networks that capture various configurations in a controlled way where the ground truth partitions and the relevant dimensions are known beforehand. Therefore, it is possible to evaluate the detection accuracy of compared algorithms using a supervised metric. To this end, we rely on the normalized mutual information (NMI) [40], a well-known metric that evaluates the detection accuracy by calculating the similarity between the reference partition (that is, the ground truth) and the obtained partition (that is, the partition generated by the community detection algorithm). The more similar the two partitions, the larger the values of NMI. The NMI achieves its maximum, which is 1, when the obtained partition is identical to the reference partition.

Figure 6 illustrates performance results of compared algorithms on synthetic networks, as evaluated with NMI. Recall that PMM [13] as well as SC-ML [18] and GraphFuse [10] require the number of communities to be set by the user. In our experiments, the target number of communities was set to the real number of generated communities. Furthermore, since PMM [13], SC-ML [18], and GraphFuse [10] are parameter-laden, we have tried several values for each input parameter. Specifically, for PMM [13], we selected the number of structural features in [5, 14] with 1 graded increments. For SC-ML [18] and GraphFuse [10], the values of the regularization parameter and the sparsity penalty factor were both taken from [0, 1] with 0.1 graded increments. In addition, due to the nondeterministic nature of all competing approaches, including ours, we performed ten repetitive executions for each selected parameter value. Based on the obtained partitions, the average $\pm$ standard deviation NMI value is reported. In fact, we should point out that our approach is nondeterministic due to its parallel processing of the nodes list and the random memberships selection when more than a membership label is equally dominating the neighborhood of a visited (processed) node. However, as the knowledgeable reader will observe from the results, the impact that poses this nondeterministic aspect to MCDA is not significant. It is also worth noting that, for each run of ensemble clustering, the final partition is identified using the best results reported by one of the three consensus techniques (CSPA, HGPA, and MCLA) [14]. Finally note that Fig. 6 does not show the results of GraphFuse on Dataset 3, Dataset 4, and Dataset 5. This is because the running time of GraphFuse is very high and we were not able to obtain an output from this algorithm within a reasonable time frame. In our experiments, we observed that its takes more than 48 h for GraphFuse to produce a partition for a network with more than 1000 nodes.

As can be seen from Fig. 6, with the exception of ensemble clustering, all approaches were able to recover the real partition of the first dataset, thanks to the shared community structure where each dimension helps in compensating the missing information from the other dimensions. Ensemble clustering on the other hand did not succeed in unfolding the real partition of this network despite the shared partition across its dimensions. We argue that such a result can be explained by the fact that this approach does not properly handle the structural variations of the network's dimensions since they are explored separately. This can be confirmed by the sub-par performance of this approach on the remaining datasets, and especially, when the number of relevant dimensions is low.

In the other datasets, MCDA reports a better performance and consistent results despite the structural variations across the networks' dimensions. This is true even for the datasets where the number of irrelevant dimensions is high (specifically Dataset 4 and Dataset 5). Such a performance can be explained by the ability of the proposed weighting schemes to efficiently discriminate between same community neighbors when assigning new node memberships. The expressed constraints do indeed help in guiding the propagation process towards the best possible partition of the network. Overall, SC-ML and PMM obtain quite comparable results. However, these two approaches were not as successful as MCDA in satisfactorily partitioning the last four datasets (that is, Datasets 2–5). In addition, both SC-ML and PMM, as well as ensemble clustering, report a slight performance decrease on Dataset 5 in comparison with Dataset 4. Recall that Dataset 5 was constructed by adding 25 irrelevant dimensions to Dataset 4. Therefore, the presence of a larger number of irrelevant dimensions explains this performance drop. In contrast, our approach

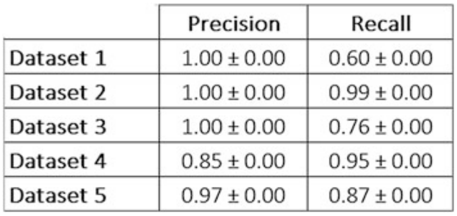


Fig. 7 Accuracy of selected dimensions identified by MCDA

	Precision	Recall
Dataset 1	1.00 ± 0.00	0.60 ± 0.00
Dataset 2	1.00 ± 0.00	0.99 ± 0.00
Dataset 3	1.00 ± 0.00	0.76 ± 0.00
Dataset 4	0.85 ± 0.00	0.95 ± 0.00
Dataset 5	0.97 ± 0.00	0.87 ± 0.00

achieves a stable performance. Our experiments suggest that, although MCDA is a nondeterministic approach, the algorithm yields quite comparable partitions on all synthetic networks (except Dataset 2), as can be seen from the low standard deviation values (see Fig. 6). For Dataset 2, however, we have observed that our approach sometimes merges two communities into a larger community. This is likely due to the high ratio of outgoing links between these two communities.

We now turn our attention to the evaluation of the performance of the relevant dimensions selection procedure of MCDA. Note that, in this evaluation, we did not consider the compared algorithms, as they do not offer any mechanisms that permit explicit selection of the relevant dimensions associated with the identified communities. In our experiments, we used precision and recall to evaluate the accuracy of the selected dimensions identified by MCDA. Specifically, for each community, precision is the ratio of the number of real relevant dimensions being selected to the number of selected dimensions. Recall is the number of real relevant dimensions selected divided by the actual number of real relevant dimensions. The reported value of a resulting partition is the average of all detected communities.

Figure 7 reports the obtained precision and recall values of the selected dimensions for the partitions identified by our algorithm. As can be seen, the high precision values reported by MCDA confirm its performance in disregarding irrelevant dimensions even for low values of o_r which characterize specifically Dataset 4 and Dataset 5. The same observation holds true for the recall values (with the exception of Dataset 1). This demonstrates the capacity of our approach in selecting the real relevant dimensions. For Dataset 1, however, we observed that MCDA, indeed, disregards some relevant dimensions with a relatively low density, compared to the other dimensions. This is expected since our relevant dimensions' detection procedure assumes the existence of irrelevant dimensions, which is not the case with this network.

3.1.3 Outlier Immunity

The aim of this set of experiments was to test the effect of the presence of outliers on the performance of MCDA. To this end, we generated five artificial networks with a varied number of outliers. Specifically, for each network, we fixed the total

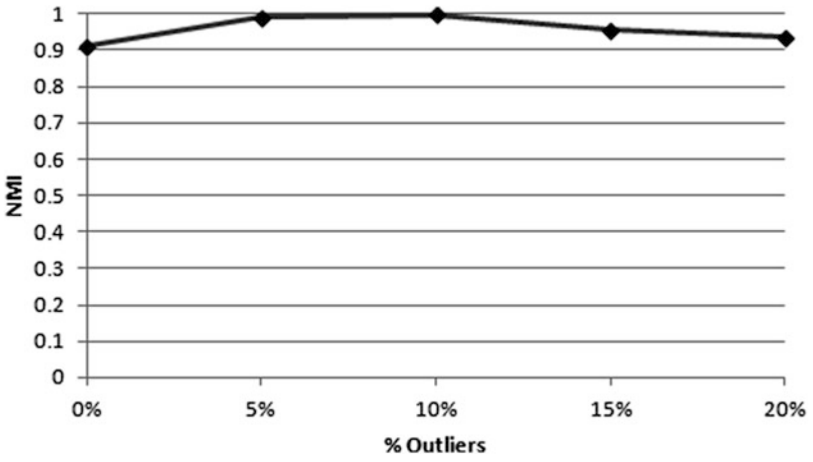


Fig. 8 Performance of MCDA in the presence of outliers

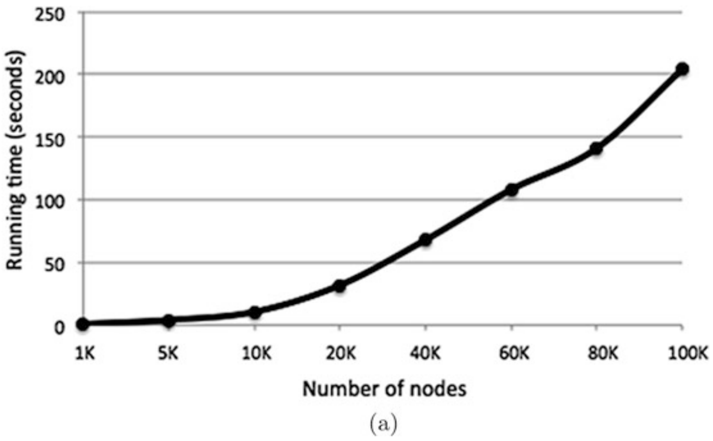
number of nodes $n = 3000$, the number communities $K = 5$, the number of dimension $o = 50$, and the average community dimensionality $o_r = 6\%$ of o, that is, only 3 dimensions out of 50 dimensions were relevant, while the remaining 47 dimensions were noisy dimensions that did not contain any community structures. In each generated network, the percentage of outliers PO varied from 0 to 20% of n. Figure 8 illustrates the community detection results (evaluated with NMI) of MCDA on these networks. Note that we did not consider competing algorithms (PMM, SC-ML, GraphFuse, ensemble clustering) in this experiment because these methods are not able to handle outliers.

As we can see from Fig. 8, MCDA displays consistent performance and is less sensitive to the percentage of outliers in the investigated networks. This result suggests that, in difficult cases, that is, when the average cluster dimensionality is low ($o_r = 6\%$ of o in our experiment), the proposed algorithm withstands the increasing value of PO and maintains high community detection accuracy. In our experiments, we found that the outlier handling procedure of MCDA removed most outlier nodes and did not miss any densely connected nodes, at the expense of also selecting a few outlier nodes. This is not necessarily an error, since outlier nodes were randomly placed throughout the entire network dimensions, and it is probable that some of them have actually been placed inside dense regions. Under these circumstances, it is possible that few outlier nodes will receive high outlier score values and consequently be considered as normal nodes that may lie within a community. This is not a serious problem, as including a few outlier nodes has no major effect on MCDAs community detection process, whereas missing a few community nodes might mean missing a substantial proportion of information. Our claim is supported by the excellent accuracy of MCDA, as depicted in Fig. 8.

3.1.4 Scalability Experiments

In this section, we study the scalability of MCDA with increasing network size and dimensionality. It is important to note that we could not compare the running time of competing algorithms due to the fact that some of them, such as ensemble clustering and GraphFuse, require a lot of time and memory resources to provide an output. Furthermore, the algorithms considered in the comparison are implemented in various platforms (we have used the original implementation of these algorithms as provided by their respective authors). This makes the running time comparison not obvious and less fair.

Scalability with Respect to the Network Size Figure 9a illustrates the running time of MCDA on networks with four dimensions while the number of nodes *n*



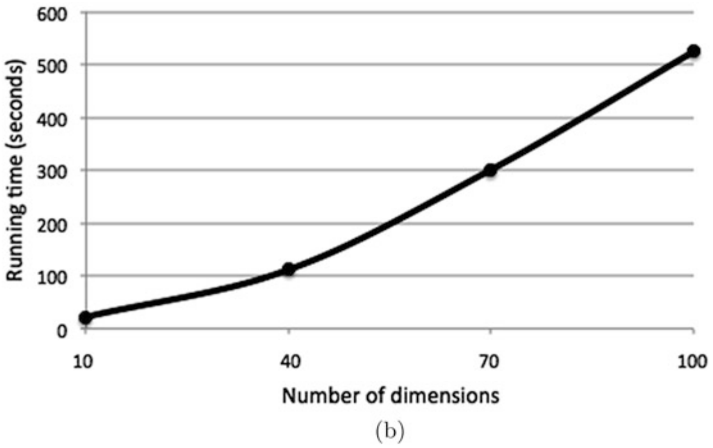


Fig. 9 Scalability experiments. (**a**) Scalability with the network size. (**b**) Scalability with the network dimensionality

varies from 1000 to 100,000. We fixed the number of relevant dimensions to two. The number of communities in each network is equal to $n/100$. The curve in Fig. 9a exhibits a quadratic behavior as the number of nodes increases. For a network with 100,000, the algorithm identifies communities and their relevant dimensions usually in less than 4 min. For a parameter free algorithm that does not require any human intervention, we believe that the execution time of MCDA is reasonable.

Scalability with Respect to the Network Dimensionality In Fig. 9b we see that MCDA scales linearly with the increase in the network dimensionality. The results presented are for networks with 3000 nodes grouped in 10 communities. The number of dimensions o varies from 10 to 100. The average community dimensionality o_r is equal to 40% of the whole dimensionality o. The results depicted in Fig. 9b illustrate that our algorithm scales well as the dimensionality of the network increases.

Finally, it is worth noting that in all the scalability experiments, the quality of the results returned by MCDA is similar to that presented in the previous subsection. Overall, our algorithm reports, on average (throughout all networks used to study the scalability), a NMI value of 0.94, pointing to accurate results.

3.2 Experiments on Real Networks

We now evaluate the performance of our approach on real networks. To this end, we selected five networks with various configurations:

DBLP1 and DBLP2 Networks These are two co-authorship three-dimensional networks that were constructed from the DBLP online database [10]. Each network dimension represents a different kind of interactions. Specifically, the first dimension represents citations between authors. The second dimension describes co-authorship relationships between authors. Finally, the third dimension connects two authors if they publish papers that share three or more terms in their titles or abstracts. The DBLP1 network captures a subset of 1230 authors of the DBLP database whereas the DBLP2 network represents 3090 authors. It is worth noting that both DBLP1 and DBLP2 networks are directed. In the context of our experiments, however, we ignored the direction of edges.

Aarhus Computer Science Department Social Network This is an unweighted and undirected five-dimensional social network capturing interactions between the employees of the computer science department at Aarhus University [41]. The dataset consists of 61 employees (admin staff, professors, associates, Ph.D. students, and post-doctoral researchers) belonging to eight workgroups and interacting in five different dimensions: launch together, friendship on Facebook, co-authorship, leisure time together, and finally work together.

Caenorhabditis Elegans Genetic Interactions Network This is a multidimensional network of 3879 nodes and six dimensions each of which capture a different type of genetic interactions of the Caenorhabditis elegans nematode roundworm [42, 43]. The interaction types being considered are: direct interaction, physical association, additive genetic interaction defined by inequality, suppressive genetic interaction defined by inequality, association, and finally colocalization.

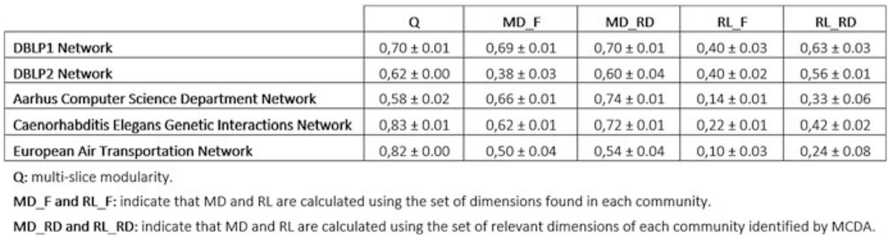
European Air Transportation Network This is a multidimensional network of 37 dimensions corresponding to different airline carriers operating between 450 European airports [44]. The network is characterized by the presence of several sparse dimensions with a relatively low contribution of links.

Note that neither the number of communities nor their relevant dimensions are known beforehand with these networks. Therefore we considered unsupervised metrics to evaluate the performance of compared algorithms. In contrast to supervised metrics, which rely on a reference partitioning, an unsupervised metric is used when the quality of the identified communities is evaluated in terms of measurable quantities obtained from the available data.

In our experiments, we considered both local and global unsupervised metrics. Generally, as suggested in [45], local metrics are based on the assumption that a community has weak interactions with its neighboring vertices. The evaluation of a community can thus be isolated from the rest of the network. On the other hand, global metrics consider the quality of a community internally (through within-community links) *and* with respect to its interactions with the other communities. In what follows, we provide a high level description of the local and global metrics that we have used. More details about these metrics can be found in their original papers referred to below.

To evaluate the performance of community detection, we relied on the multi-slice modularity Q [19] as a global metric. The multi-slice modularity Q returns values in [0, 1] such that the highest values suggest a better separation of densely connected communities. As a local metric, we have used the multidimensional community density of links MD [15] and the redundancy of links RL [12]. For a given community C_k, $MD(C_k)$ measures the ratio of the number of edges found in C_k to the maximum possible number of edges for that community. On the other hand, $RL(C_k)$ captures the ratio of the number of edges connecting adjacent nodes, in at least two dimensions, to the theoretical maximum number of edges between all connected pairs in C_k. Both $MD(C_k)$ and $RL(C_k)$ return values in [0,1] such that the largest values indicate, respectively, a higher connectedness within C_k and a higher similarity between the induced unidimensional subgraphs by C_k across G's dimensions. The reported values of a resulting partition are the average of all detected communities.

Finally, we have further used local metrics to evaluate the accuracy of our proposed selection procedure for the relevant dimensions. Specifically, we compared the computed values on both the subset of selected relevant dimensions D_k and the subset of dimensions found between nodes belonging to a single community, which we refer to as F_k. Note that $D_k \subseteq F_k$; therefore, we aim at selecting a subset D_k of F_k which induces an improvement in the originally estimated values of both $MD(C_k)$ and $RL(C_k)$ with respect to F_k.



	Q	MD_F	MD_RD	RL_F	RL_RD
DBLP1 Network	0,70 ± 0.01	0,69 ± 0.01	0,70 ± 0.01	0,40 ± 0.03	0,63 ± 0.03
DBLP2 Network	0,62 ± 0.00	0,38 ± 0.03	0,60 ± 0.04	0,40 ± 0.02	0,56 ± 0.01
Aarhus Computer Science Department Network	0,58 ± 0.02	0,66 ± 0.01	0,74 ± 0.01	0,14 ± 0.01	0,33 ± 0.06
Caenorhabditis Elegans Genetic Interactions Network	0,83 ± 0.01	0,62 ± 0.01	0,72 ± 0.01	0,22 ± 0.01	0,42 ± 0.02
European Air Transportation Network	0,82 ± 0.00	0,50 ± 0.04	0,54 ± 0.04	0,10 ± 0.03	0,24 ± 0.08

Q: multi-slice modularity.

MD_F and RL_F: indicate that MD and RL are calculated using the set of dimensions found in each community.

MD_RD and RL_RD: indicate that MD and RL are calculated using the set of relevant dimensions of each community identified by MCDA.

Fig. 10 Performance of MCDA in terms of Q, MD, and RL on real networks

Since we don't have any prior knowledge about the number of communities in real networks, we do not consider competing algorithms that require the number of communities as an input parameter (specifically, PMM, SC-ML, and GraphFuse). On the other hand, we have evaluated ensemble clustering on real networks since we used the Louvain algorithm [21] as a base detector. In fact, the Louvain approach is parameter free and thus allows the application of ensemble clustering on real networks without tuning any input parameters including the targeted number of communities to be identified. In our experiments, however, we have observed that the results of ensemble clustering using the three consensus techniques, CSPA, HGPA, and MCLA proposed in [14], are less competitive and almost similar, in terms of general performance, to the results reported on synthetic networks. Due to the fact that a comparison with ensemble clustering does not provide any additional insight to what was already observed on synthetic networks, we decided to not show the results of ensemble clustering on real networks.

Figure 10 summarizes the performance results of MCDA on real networks. As can be seen from the figure, the performance of the approach, as evaluated with the three metrics Q, MD, and RL, suggests the extraction of statistically relevant community structures. More specifically, with respect to the multi-slice modularity Q, MCDA obtains large values on the five real datasets, which confirms its ability to recover dense and well-separated communities. This claim is supported by the high multidimensional community density of link MD values obtained on the subsets of found dimensions F_k.

With respect to the redundancy of links RL, MCDA detects communities that are similar across their relevant dimensions. This is especially the case of the DBLP networks where the interactions between authors of the same community tend to be correlated. In fact, it is expected that researchers who co-author papers (first dimension) tend to specialize in the same research tracks, which could explain the shared words on the abstract of their papers (third dimension) and the citation of each other's work (second dimension). This correlation of activity across the densely connected communities suggests a track oriented organization in which each group of researchers specializes in a single research topic. The same observation holds true for the communities recovered for the Caenorhabditis Elegans Genetic Interactions network which could suggest the presence of functional modules that may characterize some biological phenomenon or support new hypotheses that can be verified using existing biological expertise. MCDA, on the other hand, reports relatively low values for the redundancy RL on the European air transportation network. This low correlation of activity might be due to the fact that low-cost air carriers use a different business model by avoiding air fares between airports covered by major airlines [44].

The performance of the procedure adopted by MCDA for the selection of relevant dimensions can also be demonstrated through the improvements in the estimated values of MD and RL on the subsets F (dimensions found within each community) and RD (identified relevant dimensions of each community). As depicted in Fig. 10, on average, the values of MD and RL improve by 0.1 and 0.18, respectively, on the selected relevant dimensions (see the values of MD_F vs MD_RD and RL_F vs RL_RD). Such a performance confirms the ability of MCDA in identifying the real relevant dimensions that support the formation of the detected communities.

4 Conclusion

In this paper, we have addressed the problem of community detection in multidimensional networks. The proposed approach tackles some of the limitations of existing algorithms namely, the parameter tuning and the inability to recover the most relevant dimensions associated with the detected communities. The experimental evaluation suggests that MCDA offers a higher accuracy and usability than the compared approaches. In addition, the capacity to disregard non-informative dimensions contributes valuable input that helps in understanding the key interaction drivers among groups. We believe that the selected relevant dimensions can also benefit other tasks such as network compression, representation learning, and collaborative filtering. Finally, we believe that our approach can successfully be applied for the discovery of time evolving communities in dynamic networks. This can be achieved by considering the snapshots or alternatively the time intervals of the network's time-ordered sequence as distinct dimensions. Further investigation is needed in this direction.

Acknowledgements This work is supported by research grants from the Natural Sciences and Engineering Research Council of Canada (NSERC).

References

1. Battiston, F., Nicosia, V., Latora, V.: Structural measures for multiplex networks. Phys. Rev. E **89**(3), 032804 (2014)
2. Nicosia, V., Latora, V.: Measuring and modelling correlations in multiplex networks (2014). Preprint, arXiv: 1403.1546
3. Berlingerio, M., Coscia, M., Giannotti, F., Monreale, A., Pedreschi, D.: Multidimensional networks: foundations of structural analysis. World Wide Web **16**(5–6), 567–593 (2013)

4. Cellai, D., López, E., Zhou, J., Gleeson, J.P., Bianconi, G.: Percolation in multiplex networks with overlap. Phys. Rev. E **88**(5), 052811 (2013)
5. Cozzo, E., Banos, R.A., Meloni, S., Moreno, Y.: Contact-based social contagion in multiplex networks. Phys. Rev. E **88**(5), 050801 (2013)
6. Amelio, A., Pizzuti, C.: A cooperative evolutionary approach to learn communities in multilayer networks. In: Parallel Problem Solving from Nature – PPSN XIII, pp. 222–232. Springer, Cham (2014)
7. Tang, L., Wang, X., Liu, H.: Uncovering groups via heterogeneous interaction analysis. In: 9th IEEE International Conference on Data Mining (ICDM), pp. 503–512 (2009)
8. Boutemine, O., Bouguessa, M.: Mining community structures in multidimensional networks. ACM Trans. Knowl. Discov. Data (TKDD) **11**(4), 51 (2017)
9. Boden, B., Gunnemann, S., Hoffmann, H., Seidl, T.: Mining coherent subgraphs in multi-layer graphs with edge labels. In: 18th ACM SIGKDD International Conference on Knowledge Discovery and Data mining (KDD), pp. 1258–1266 (2012)
10. Papalexakis, E.E., Akoglu, L., Ience, D.: Do more views of a graph help? Community detection and clustering in multi-graphs. In: 16th International Conference on Information Fusion (FUSION), pp. 899–905 (2013)
11. Cai, D., Shao, Z., He, X., Yan, X., Han, J.: Mining hidden community in heterogeneous social networks. In: 3rd ACM International Workshop on Link Discovery (LinkKDD), pp. 58–65 (2005)
12. Berlingerio, M., Coscia, M., Giannotti, F.: Finding and characterizing communities in multi-dimensional networks. In: International Conference on Advances in Social Networks Analysis and Mining (ASONAM), pp. 490–494. IEEE, Piscataway (2011)
13. Tang, L., Wang, X., Liu, H.: Community detection via heterogeneous interaction analysis. Data Min. Knowl. Discov. **25**(1), 1–33 (2012)
14. Strehl, A., Ghosh, J.: Cluster ensembles—a knowledge reuse framework for combining multiple partitions. J. Mach. Learn. Res. **3**, 583–617 (2003)
15. Berlingerio, M., Pinelli, F., Calabrese, F.: ABACUS: frequent pAttern mining-BAsed community discovery in mUltidimensional networkS. Data Min. Knowl. Discov. **27**(3), 294–320 (2013)
16. Borgelt, C.: Efficient implementations of Apriori and Eclat. In: ICDM Workshop on Frequent Itemset Mining Implementations (FIMI) (2003)
17. Tang, W., Lu, Z., Dhillon, I.S.: Clustering with multiple graphs. In: 9th IEEE International Conference on Data Mining (ICDM), pp. 1016–1021 (2009)
18. Dong, X., Frossard, P., Vandergheynst, P., Nefedov, N.: Clustering on multi-layer graphs via subspace analysis on Grassmann manifolds. IEEE Trans. Signal Process. **62**(4), 905–918 (2014)
19. Mucha, P.J., Richardson, T., Macon, K., Porter, M.A., Onnela, J.P.: Community structure in time-dependent, multiscale, and multiplex networks. Science **328**(5980), 876–878 (2010)
20. Carchiolo, V., Longheu, A., Malgeri, M., Mangioni, G.: Communities unfolding in multislice networks. In: Complex Networks, pp. 187–195. Springer, Berlin (2011)
21. Blondel, V.D., Guillaume, J.L., Lambiotte, R., Lefebvre, E.: Fast unfolding of communities in large networks. J. Stat. Mech. Theory Exp. **2008**(10), P10008 (2008)
22. De Domenico, M., Lancichinetti, A., Arenas, A., Rosvall, M.: Identifying modular flows on multilayer networks reveals highly overlapping organization in interconnected systems. Phys. Rev. X **5**(1), 011027 (2015)
23. Rosvall, M., Bergstrom, C.T.: Maps of random walks on complex networks reveal community structure. Proc. Natl. Acad. Sci. **105**(4), 1118–1123 (2008)
24. Dunlavy, D.M., Kolda, T.G., Kegelmeyer, W.P.: Multilinear algebra for analyzing data with multiple linkages. In: Kepner, J., Gilbert, J. (eds.) Graph Algorithms in the Language of Linear Algebra. Fundamentals of Algorithms, pp. 85–114. SIAM, Philadelphia (2011)
25. Kuncheva, Z., Montana, G.: Community detection in multiplex networks using locally adaptive random walks. In: Proceedings of the 2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2015, pp. 1308–1315. ACM, New York (2015)

26. Bouguessa, M.: An unsupervised approach for identifying spammers in social networks. In: Tools with Artificial Intelligence (ICTAI), 2011 23rd IEEE International Conference on, pp. 832–840. IEEE, Piscataway (2011)
27. Ji, Y., Wu, C., Liu, P., Wang, J., Coombes, K.R.: Applications of beta-mixture models in bioinformatics. Bioinformatics **21**(9), 2118–2122 (2005)
28. Bouguila, N., Ziou, D., Monga, E.: Practical Bayesian estimation of a finite beta mixture through Gibbs sampling and its applications. Stat. Comput. **16**(2), 215–225 (2006)
29. Ma, Z., Leijon, A.: Beta mixture models and the application to image classification. In: 16th IEEE International Conference on Image Processing, pp. 2045–2048 (2009)
30. Boutemedjet, S., Ziou, D., Bouguila, N.: Model-based subspace clustering of non-Gaussian data. Neurocomputing **73**(10), 1730–1739 (2010)
31. Smyth, P.: Model selection for probabilistic clustering using cross-validated likelihood. Stat. Comput. **10**(1), 63–72 (2000)
32. Schwarz, G., et al.: Estimating the dimension of a model. Ann. Stat. **6**(2), 461–464 (1978)
33. Figueiredo, M.A.T., Jain, A.K.: Unsupervised learning of finite mixture models. IEEE Trans. Pattern Anal. Mach. Intell. **24**(3), 381–396 (2002)
34. Bezdek, J.C.: Pattern Recognition with Fuzzy Objective Function Algorithms. Springer Science & Business Media, New York (2013)
35. Raghavan, U.N., Albert, R., Kumara, S.: Near linear time algorithm to detect community structures in large-scale networks. Phys. Rev. E **76**(3), 036106 (2007)
36. Barber, M.J., Clark, J.W.: Detecting network communities by propagating labels under constraints. Phys. Rev. E **80**(2), 026129 (2009)
37. Leung, I.X., Hui, P., Lio, P., Crowcroft, J.: Towards real-time community detection in large networks. Phys. Rev. E **79**(6), 066107 (2009)
38. Liu, X., Murata, T.: Advanced modularity-specialized label propagation algorithm for detecting communities in networks. Phys. A Stat. Mech. Appl. **389**(7), 1493–1500 (2010)
39. Condon, A., Karp, R.M.: Algorithms for graph partitioning on the planted partition model. Random Struct. Algorithms **18**(2), 116–140 (2001)
40. Manning, C.D., Raghavan, P., Schütze, H., et al.: Introduction to Information Retrieval, vol. 1. Cambridge University Press, Cambridge (2008)
41. Magnani, M., Micenkova, B., Rossi, L.: Combinatorial analysis of multiple networks. Preprint, arXiv: 1303.4986 (2013)
42. De Domenico, M., Nicosia, V., Arenas, A., Latora, V.: Structural reducibility of multilayer networks. Nat. Commun. **6**, 6864 (2015)
43. Stark, C., Breitkreutz, B.-J., Reguly, T., Boucher, L., Breitkreutz, A., Tyers, M.: BioGRID: a general repository for interaction datasets. Nucleic Acids Res. **34**(1), D535–D539 (2006)
44. Cardillo, A., Gómez-Gardeñes, J., Zanin, M., Romance, M., Papo, D., del Pozo, F., Boccaletti, S.: Emergence of network features from multiplexity. Sci. Rep. **3**, 1344 (2013)
45. Loe, C.W., Jensen, H.J.: Comparison of communities detection algorithms for multiplex. Phys. A Stat. Mech. Appl. **431**, 29–45 (2015)

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 $\chi(\nu, \xi)$

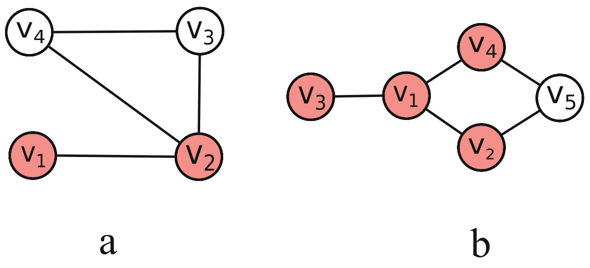


Fig. 1 Examples of shapes in graphs

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.

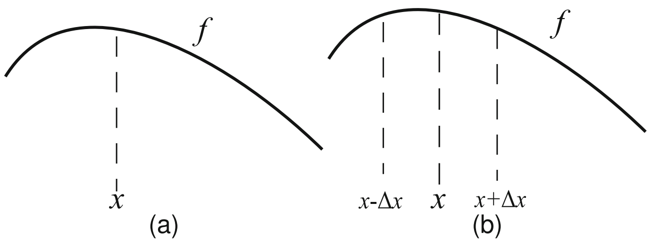


Fig. 2 (**a**) Approximating derivative of f at x, (**b**) discretising f into three points: $x - \Delta x$, x, and $x + \Delta x$

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.

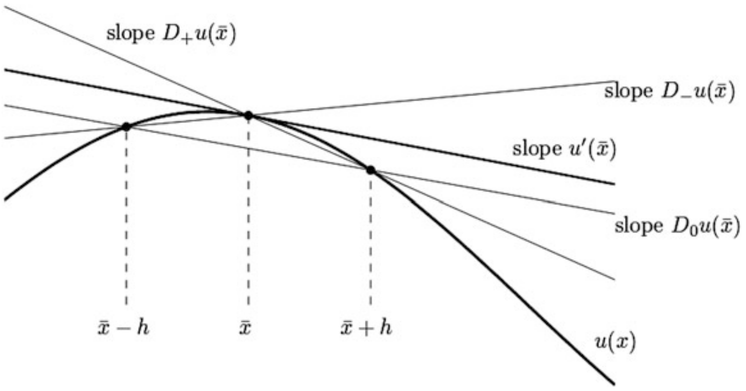


Fig. 3 Approximating the derivative of $f(x)$ using Taylor series

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

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

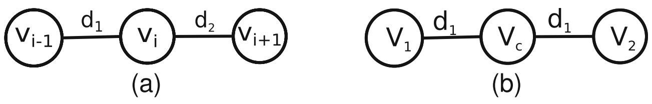


Fig. 4 (**a**) Example graph with three nodes, (**b**) derivatives of f at v_c which has two neighbours

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 $d_i(w) = \frac{d_i}{w}$.

The second derivative according to the Taylor series:

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

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:

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

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

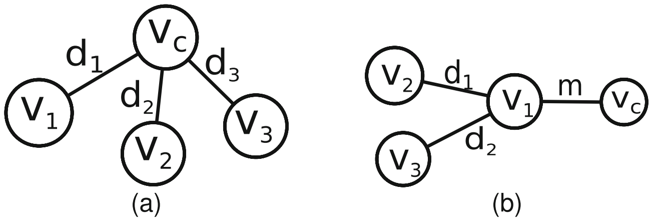


Fig. 5 (**a**) Approximating derivatives of F at v_c which has three neighbours, (**b**) non-central nodes

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

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:

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

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

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

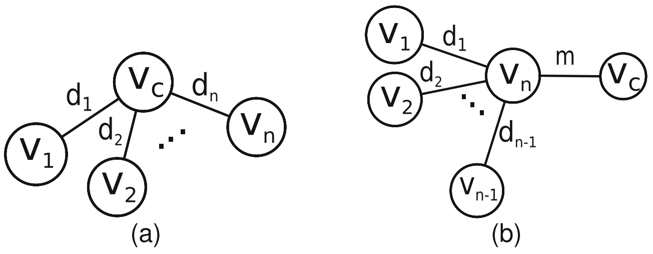


Fig. 6 (**a**) Approximating derivative of F at node v_c with n neighbours, (**b**) approximating derivative of F at the non-central node v_c

These equations result into the following system of equation:

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

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:

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

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:

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

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 $C$
```

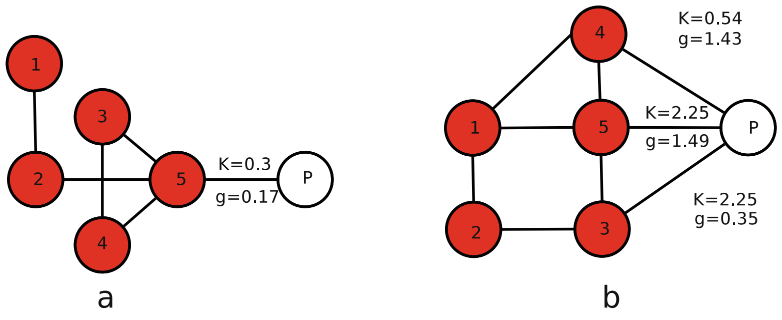


Fig. 7 In both (**a**) and (**b**), shape χ consists of v_1, v_2, v_3, v_4, and v_5 and v_p is the candidate for the inclusion. In (**a**), values of curvature and gradient from v_5 towards v_p are shown. In (**b**), values of curvature and gradient from v_3, v_4, and v_5 towards v_p are shown

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

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

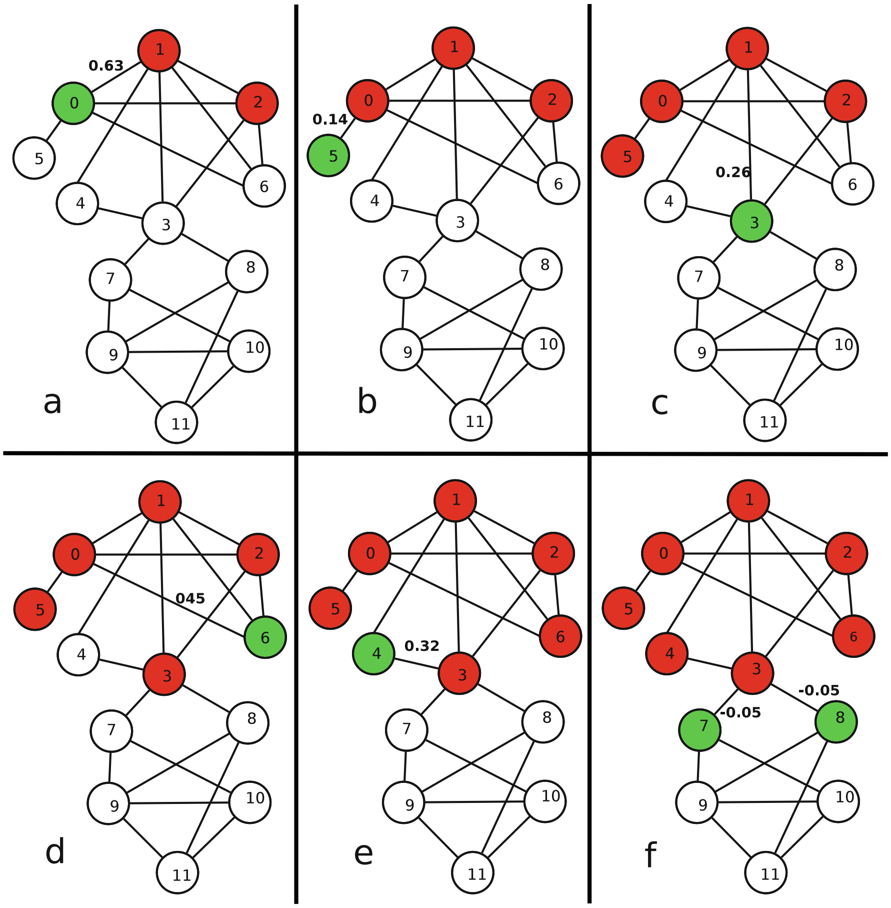


Fig. 8 The red nodes show the current community and the green nodes are candidates for inclusion. The number on the edges shows the velocity from a node to the candidate. When the velocity towards all neighbouring nodes is negative, the algorithm stops

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

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

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).

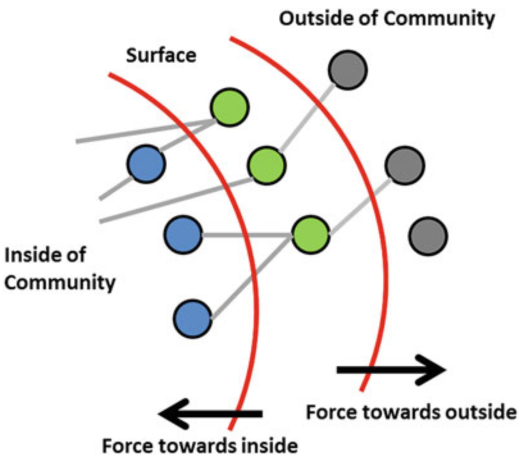


Fig. 9 Surface of a community and its binding forces

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

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

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

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

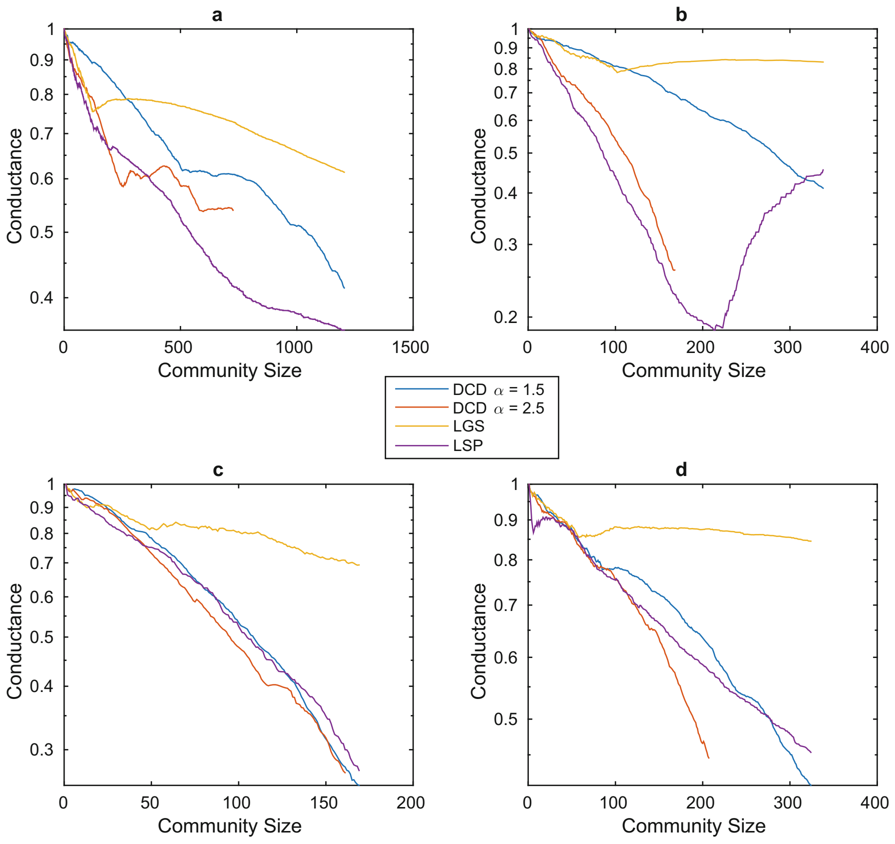


Fig. 10 Conductance plot for different methods and different starting nodes in FB-JHK. Initial seed: (**a**) 2645, (**b**) 3229, (**c**) 3554, and (**d**) 3718

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

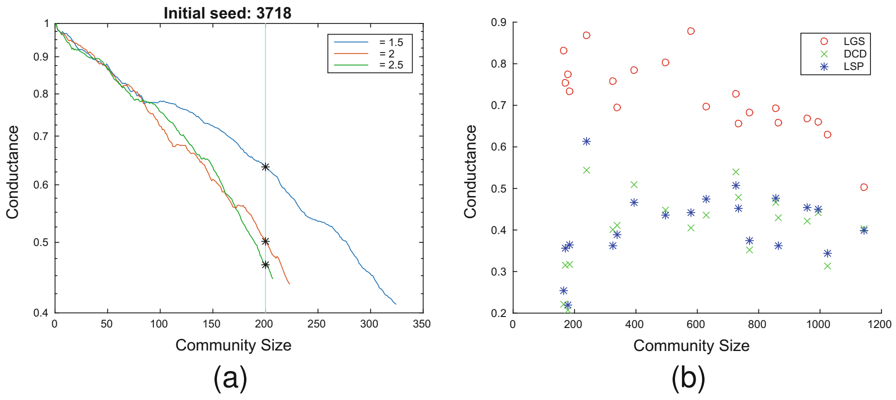


Fig. 11 (**a**) Effect of α on finding local communities around a seed node in FB-JHK, (**b**) conductance of the different detected communities in FB-JHK where $\alpha = 1.5$

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

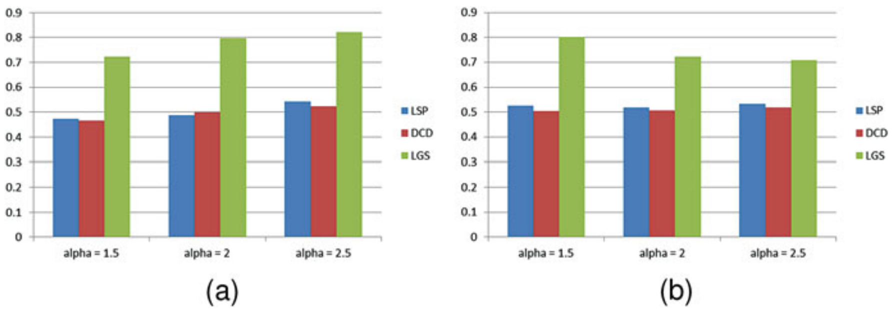


Fig. 12 (**a**) Average conductances of the communities in FB-JHK, (**b**) average conductances of the communities in FB-CALTC

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

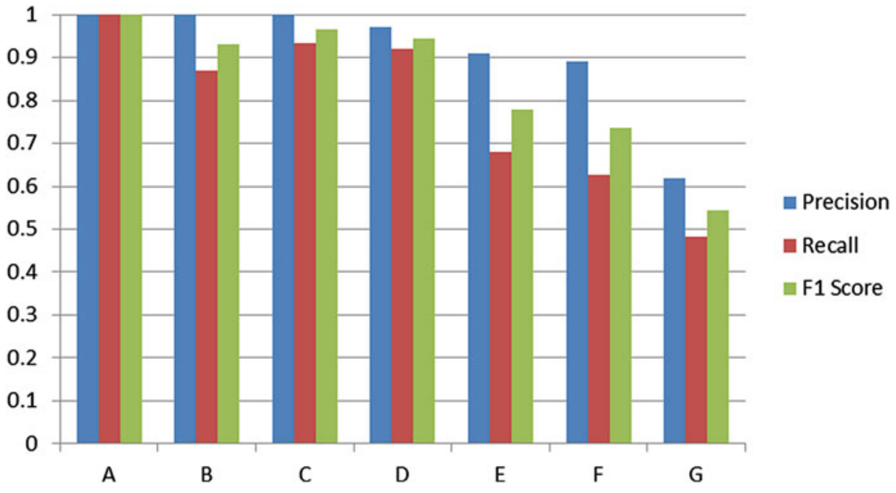


Fig. 13 Precision, recall, and F1 score for each scenario in Table 5

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)

Graph Clustering Based on Attribute-Aware Graph Embedding

Esra Akbas and Peixiang Zhao

Abstract Graph clustering is a fundamental problem in graph mining and network analysis. To group vertices of a graph into a series of densely knitted clusters with each cluster being well-separated from all the others, classic methods primarily consider the mere graph structure information in modeling and quantifying the proximity or distance of vertices for graph clustering. However, with the proliferation of rich, heterogeneous attribute information widely available in real-world graphs, such as user profiles in social networks, and GO (Gene Ontology) terms in protein interaction networks, it becomes essential to combine both structure and attribute information of graphs towards yielding better-quality clusters. In this chapter, we propose a new graph embedding approach for *attributed graph clustering*. We embed each vertex of a graph into a continuous vector space within which the local structure and attribute information surrounding the vertex can be jointly encoded in a unified, latent representation. Specifically, we quantify the vertex-wise attribute proximity into edge weights and leverage a group of truncated, attribute-aware random walks to learn the latent representations of vertices. This way, the challenging attributed graph clustering problem can be cast into the traditional problem of multidimensional data clustering, which has admitted efficient and cost-effective solutions. We apply our attribute-aware graph embedding algorithm in a series of real-world and synthetic attributed graphs and networks. The experimental studies demonstrate that our proposed method significantly outperforms the state-of-the-art attributed graph clustering techniques in terms of both clustering effectiveness and efficiency.

E. Akbas
Oklahoma State University, Stillwater, OK, USA
e-mail: eakbas@cs.okstate.edu

P. Zhao (✉)
Florida State University, Tallahassee, FL, USA
e-mail: zhao@cs.fsu.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_5

1 Introduction

Graph clustering, also referred to as *community detection*, is a fundamental data mining problem for exploring and understanding graphs and networked data [10, 26]. The objective of classic graph clustering is to partition vertices of a graph based on its topological structure information, such as vertex connectivity and distance, in order to identify densely connected subgraphs or communities. However, in addition to the interlinked graph structures, real-world graphs and information networks have witnessed an abundant amount of rich, attributive information affiliated with vertices that represent the key characteristics and properties of entities [15, 36]. This gives rise to a new, complicated type of graphs, namely, *attributed graphs*, and as a result, new graph clustering methods need to be revamped accordingly to support the so-called *attributed graph clustering* [5, 22, 30, 32, 36, 37].

Attributed graph clustering has found a wide range of real-world applications because it has the potential to yield more informative and better-quality clustering results due in particular to a joint consideration of both attribute and structure information of the underlying graphs. Some noteworthy examples are outlined as follows:

1. In social networks such as Facebook, LinkedIn, and Google+, users and their friendship relations constitute the underlying social graphs. Additionally, each user is characterized by a series of attributes depicting one's personal profiles including age, education, occupation, location, and hobbies. Clustering social network users by tackling both their social relationships and personal profiles is particularly useful for social targeting and personalized recommendation [11, 17, 36];
2. In protein–protein interaction (PPI) networks, the proteins are often annotated with biological attributes, such as functional classifications, gene ontology (GO) terms, and gene expression profiles. Clustering PPI networks by leveraging both the network topologies and protein attributes can significantly facilitate the identification process of protein complexes and pathways [7, 14];
3. The web graph consists of web pages interweaved by hyperlinks. Each web page is also characterized by a series of attributes including URL, name, keywords, contents, tags, and so forth. When both hyperlinks and web page attributes are considered for web community discovery, the results are often more informative than those identified based solely on web structures [9, 12, 25].

Unfortunately, attributed graph clustering turns out to be significantly more challenging than the classic graph clustering problem where the mere graph structure information is adopted for clustering. The main reason is that topological structures and vertex attributes are two completely different types of information pertaining to graphs. Clustering based solely on either one type of information oftentimes leads to inconsistent, or even contradicting, graph clustering results [36]. As a consequence, the key challenge of attributed graph clustering is to meaningfully combine the structure and attribute information of graphs towards striking the right balance

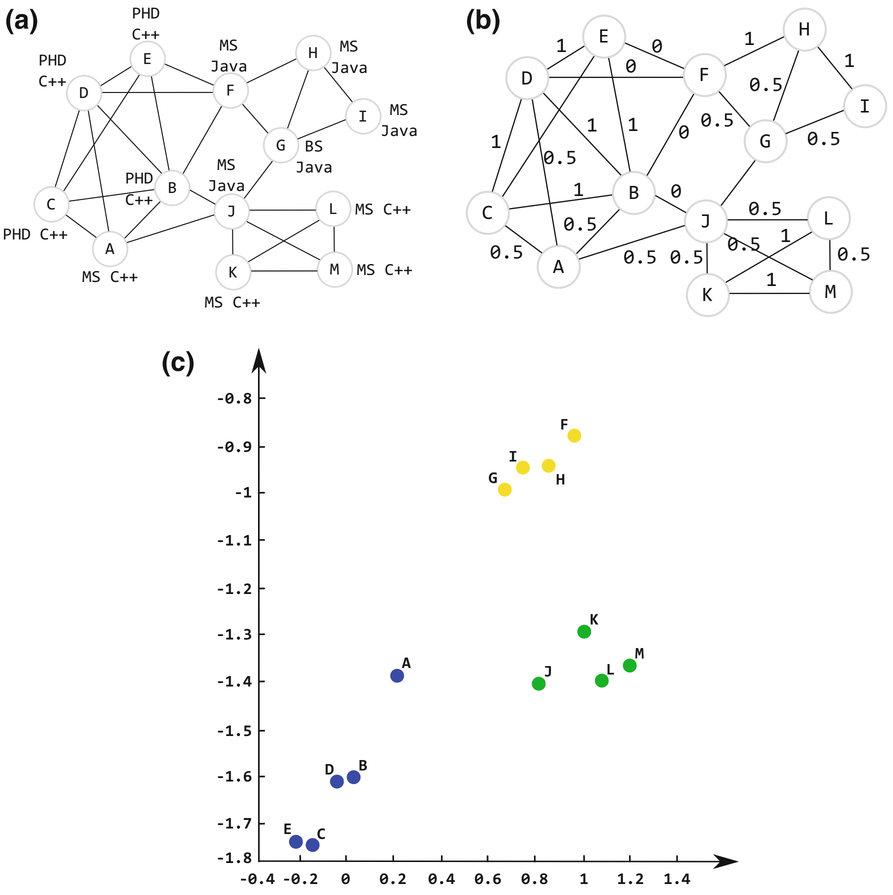


Fig. 1 The attribute-aware graph embedding on a sample attributed graph. (**a**) Presents a sample social graph G containing 13 individuals and their friendship relations. Each individual is characterized by two attributes: *education* and *favorite programming language*; (**b**) presents the transformed, weighted graph G' with vertex attribute similarity embedded as edge weights; and (**c**) presents the two-dimensional attribute-aware graph embedding, Φ, from which the latent cluster structures naturally arise

between the two distinct objectives of clustering based on either graph structures or vertex attributes, respectively.

In this chapter, we introduce a novel approach for attributed graph clustering based on both graph structure cohesiveness and vertex attribute homogeneity. The key idea is to design a unified latent representation for each vertex u of a graph such that both the graph connectivity and vertex attribute similarity within the localized region of u can be jointly encoded into a continuous vector space. Specifically, the pairwise vertex attribute similarity between u and any vertex in u's local vicinity is first quantified and embedded as the new edge weights of the graph. A series of truncated, edge weight-biased random walks originating from u are further generated to capture the local, attribute-aware structure information surrounding u. Inspired by the recent work of graph embedding [8, 23, 28], these random walks are further used to learn a latent representation, $r(u) \in \mathbb{R}^d$, of u, which lies in a continuous vector space with a relatively small number d of dimensions. This way, the salient, localized attribute and structure information of the vertices can be jointly and uniformly encoded in the continuous d-dimensional vector space. As a result, the challenging attributed graph clustering problem is cast to the traditional data clustering problem upon the d-dimensional space, in which the graph structure cohesiveness and vertex attribute homogeneity can be approximately preserved. In the end, we can apply any data clustering algorithm, e.g., k-Medoids, to accomplish the challenging task of attributed graph clustering in large-scale graphs or networks. To illustrate the key idea, we present the typical pipeline of our attributed graph clustering method in a sample graph, as shown in Fig. 1.

The main contributions of our work can be summarized as follows:

1. We propose a novel, attribute-aware graph embedding framework for the problem of attributed graph clustering. It provides a natural and principled way to encode the structure and attribute information surrounding vertices into a unified latent representation in a low-dimensional vector space, within which the graph structure cohesiveness and vertex attribute homogeneity can be well-preserved. This framework also establishes a general graph embedding approach to tackling attributed graphs in widely varying application domains.
2. We design an efficient and cost-effective graph embedding algorithm that accomplishes the transformation of an attributed graph into its vertex-based latent representations in a low-dimensional space, which are further fed as input to any data clustering method for attributed graph clustering;
3. We carry out experimental studies in a series of real-world and synthetic attributed graphs and compare our method with the state-of-the-art attributed graph clustering techniques. The experimental results demonstrate that our efficient graph embedding method outperforms other existing algorithms in terms of both graph structure cohesiveness (w.r.t. graph density) and vertex attribute homogeneity (w.r.t. entropy) of resultant graph clusters. Our scalability test shows that our algorithm exhibits excellent scalability for graph clustering, while the state-of-the-art attributed graph clustering techniques cannot scale in large graphs.

The rest of this chapter is organized as follows. In Sect. 2, we discuss the related work for attributed graph clustering and graph embedding. In Sect. 3, we formally define the problem of attributed graph clustering. We then present our attribute-aware graph embedding framework in Sect. 4 and develop the corresponding algorithm in Sect. 5. We report our experimental studies and main results in Sect. 6, followed by the conclusion in Sect. 7.

2 Related Work

In this section, we brief the related work on attributed graph clustering and graph embedding and discuss how our proposed work differs from the existing solutions.

Attributed Graph Clustering There has been a rich literature for attributed graph clustering [6]. The straightforward idea is to define some vertex-wise distance metric or similarity measure that takes into account both structure and attribute information of vertices in a graph. For instance, the differences in vertex attribute values can be quantified as distances between neighboring vertices [27]. The textual web contents and hyperlinks are also combined in a similarity measure for web page clustering [12]. SA-Cluster [36] transforms a graph into another augmented graph with new, artificial *attribute vertices* representing distinct vertex attribute values. A new *attribute edge* (u, u') is further created that links an original vertex u and a newly created attribute vertex u', if there is an attribute of u whose value is equal to u'. After this transformation, the vertices sharing the same attribute values are connected via common attribute vertices. A random walk-based distance measure is further defined upon the augmented graph in order to estimate the closeness of vertices in terms of both structure cohesiveness and attribute proximity. SA-Cluster is computationally expensive because: (1) the resultant augmented graph can be excessively large if the number of distinct vertex attribute values is high, and (2) SA-Cluster involves costly iterative matrix multiplication in random walk-based distance computation and weight tuning. Although an improved version has been proposed to support incremental computation for the random walk distance matrix [37], SA-Cluster is still hard to scale up in real-world large-scale attributed graphs.

Another stream of related work for attributed graph clustering is built primarily upon generative probabilistic models, within which graph structure and vertex attribute information is correlated to a set of shared, hidden variables of cluster membership [2, 13, 30, 32, 33]. BAGC [30, 31] is a Bayesian probabilistic model in which the cluster label of each vertex of the graph is explicitly represented as a hidden variable. A joint probability distribution is further defined and estimated over the space of all possible clusterings of the attributed graph, and a variational inference algorithm is developed to find the posterior distribution with highest probability. While a graph is considered as a dynamic system, it is assumed that community is the stable status shared by the nodes in that community. Specifically, nodes from the same community are likely to get the same amount of content propagation. CPIP [18] is a model-based method which is based on content propagation with principles of influence propagation as well as random walk. It is assumed that while a network is considered as a dynamic system, a community is considered as the stable status shared by the nodes in that community. Specifically, nodes in the same community are likely to get the same amount of content propagation. CESNA [32] assumes that clusters will generate both the graph structure and vertex attributes based on the affiliation network model [16] and a separate logistic model, respectively. An optimization problem is further formulated to probabilistically infer clusters based on both graph structure and vertex attribute information, and an efficient coordinate ascent algorithm is used to detect overlapping graph clusters from the attributed graph. Unfortunately, these probabilistic clustering methods have to undertake an optimization process for parameter likelihood estimation, which is typically very time-demanding. In addition, choosing appropriate a priori distributions in these statistical models is a nontrivial task.

Our work differs from existing attributed graph clustering solutions in that we are the first to consider the idea of attribute-aware graph embedding towards encoding both graph structure cohesiveness and vertex attribute homogeneity into a low-dimensional latent space, within which attributed graph clustering can be supported in an efficient and cost-effective way especially in large-scale attributed graphs.

Graph Embedding There have been many approaches to learning low-dimensional representations for graphs [4, 24, 29]. Inspired by the recent advancement in language modeling and deep learning [20], a series of graph embedding work that learn latent vertex representations of graphs have been proposed. DeepWalk [23] takes advantage of local structure information of vertices based on the Skip-Gram model [20] to learn the latent representations by treating random walks as the equivalent of sentences. When applied for multilabel graph classification in social networks, DeepWalk can successfully encode the global graph structure especially in the presence of missing information in graphs. Line [28] is a scalable graph embedding method that uses edge sampling for model inference. It naturally breaks the limitation of the classical stochastic gradient decent method adopted in graph embedding without compromising embedding efficiency. GraRep [8] refines DeepWalk by introducing an explicit loss function of the Skip-Gram model defined on the graph and extends Line by capturing k-step ($k > 2$) high-order information for learning the latent representations. Matrix factorization algorithms are used for optimization in GraRep.

Our work differs from existing graph embedding solutions in the following two aspects: (1) graph embedding is typically proposed and optimized for the task of graph classification, while our work is primarily designed for attributed graph clustering; (2) graph embedding considers encoding the mere graph structure into a low-dimensional space, while our work is the first to take into account attributed graphs and thus enriches the existing frameworks for attribute-aware graph embedding.

This work is an extended version of our previous conference paper [1]. We provide details of our attributed graph embedding algorithm with theoretical justifications. We also include thorough experimental results with other large-scale real-world attributed graphs and a set of synthetic attributed graph by varying several parameters including number of vertex and vertex dimensionality. We compare AA-Cluster with other state-of-the art methods. We examine the effects of our algorithm's parameter such as neighborhood distance, length and number of random walks per vertex, and window size. Moreover, we analyze the scalability of our methods and compare with others. We find that our attributed graph clustering method has excellent scalability, while others cannot scale in large graphs with giving memory error after some points.

3 Problem Formulation

Henceforth, we consider clustering graph-structured data whose vertices are affiliated with multidimensional attributes. We refer to these complicated data as *attributed graphs*, which is defined as follows.

Definition 1 (Attributed Graph) An attributed graph is a three tuple $G = (V, E, \mathcal{A})$, where V is a set of vertices, $E \subseteq V \times V$ is a set of edges, and $\mathcal{A} = \{A_1, \ldots, A_n\}$ is a set of n attributes associated with vertices from V. That is, for each $u \in V$, there is an attribute vector $\mathcal{A}(u) = (A_1(u), \ldots, A_n(u))$ associated with u, where $A_l(u)$ is the attribute value of u on the l-th attribute A_l $(1 \leq l \leq n)$.

In this chapter, we consider the attributed graphs as undirected, connected, and simple graphs, and all the vertex attributes conform to a unique multidimensional schema, $\mathcal{A}$. However, the proposed attribute-aware graph embedding framework and the attributed graph clustering algorithm can be extended with minor revision to other types of graphs with the vertex attributes conforming to heterogeneous attribute schemas. We further assume that each vertex attribute A_i has a finite set of discrete values and the number of possible values (or cardinality) of A_i is $|A_i|$. For vertex attributes with continuous or infinitely countable values, we can transform them into discrete values by either bucketization or histograms.

Given a vertex u in an attributed graph G, we denote all the neighboring vertices of u as $N_1(u) = \{v | v \in V, (u, v) \in E\}$. Analogously, we denote all the vertices that are l $(l \geq 1)$ hops away from u as $N_l(u) = \{v | v \in V, d(u, v) = l\}$, where $d(\cdot)$ is the shortest unit distance function defined upon G. If l is small, $N_l(u)$ consists of all vertices that are in the local vicinity of u. In principle, if vertices in $N_l(u)$ are densely connected and share similar vertex attributes with u, they are very likely to be in the same cluster that u belongs to.

Definition 2 (Attributed Graph Clustering) The attributed graph clustering problem is to partition a graph G into k mutually exclusive and collectively exhaustive subgraphs $G_i = (V_i, E_i, \mathcal{A})$ with an objective to obtain the following graph clustering properties:

1. **Structure closeness:** Vertices within the same clusters are closely connected with each other, while vertices in different clusters are far apart;
2. **Attribute homogeneity:** Vertices within the same clusters have similar attribute values, while vertices in different clusters differ significantly in vertex attribute values.

We note that in the classic graph clustering problem, only the first objective is optimized as the mere graph structure information is considered, while for the attributed graph clustering problem, we strive to achieve the dual objective of structure closeness and attribute homogeneity for graph clusters.

4 The Attribute-Aware Graph Embedding Framework

In this section, we will discuss our attribute-aware graph embedding framework for attributed graph clustering. Our goal is to transform every vertex u of an attributed graph into a latent, low-dimensional feature vector $f(u) \in \mathbb{R}^d$, where d is a small number for the latent dimensions. This way, both vertex attributes and local graph structure information of u are encoded within $f(u)$ in a way that vertices of the same cluster will have similar feature vectors. The distance within the latent low-dimensional space should represent a metric for evaluating the structure–attribute similarity between corresponding vertices of the attributed graph.

4.1 Vertex Attribute Embedding

Given an attributed graph G, our first step is to embed the information of vertex attribute similarity into a transformed, weighted graph $G' = (V, E; W)$, where $W : E \to \mathbb{R}_{\geq}0$. Specifically, for each edge $e = (u, v) \in E$, we assign an edge weight $w(e)$ that quantifies the vertex attribute similarity for u and v. This way, the vertex attribute information of G can be encoded into the weighted graph G' as new edge weights.

The straightforward way to quantify the multidimensional attribute similarity of two adjacent vertices u and v is based on a dimension-wise evaluation of attribute values for u and v, respectively. We define an indicator function $\mathbb{1}_{A_i}(u, v)$ for the attribute A_i of u and v as follows:

$$\mathbb{1}_{A_i}(u, v) = \begin{cases} 1 \text{ if } A_i(u) = A_i(v) \\ 0 \text{ otherwise} \end{cases}$$

Then, the vertex attribute similarity, $s_0(u, v)$, of vertices u and v can be computed as:

$$s_0(u, v) = \frac{\sum_{i=1}^{n} \mathbb{1}_{A_i}(u, v)}{n} \tag{1}$$

In an attributed graph G, it is not uncommon two adjacent vertices u and v that are within the same cluster may share few, or even no, identical vertex attribute values. In this case, u and v may be closely connected and only their structure information plays an essential role in assigning them in one cluster. However, if we solely rely on the structure information of u and v that disagree on vertex attribute values, it is still likely that u and v will be assigned to different clusters by mistake. To account for this case, we extend the computation of the vertex attribute similarity by taking into consideration the neighboring vertices of u and v, respectively. This way, if u and v share few or no common vertex attribute values, the vertices within their vicinity may still hold identical or similar vertex attribute values, considering that u and v belong to the same cluster. Formally, we consider all the nearby vertices of u in $N_l(u)$ which are l hops away from u. For each vertex attribute $A_i (1 \leq i \leq n)$, we maintain a histogram vector, $H_{A_i}(u)$, with a total number of $|A_i|$ entries, each of which corresponds to a possible value $a_t \in \text{Domain}(A_i)$, and maintains the value as:

$$H_{A_i}(u)[t] = \frac{|\{v|v \in N_l(u), A_i(v) = a_t\}|}{|N_l(u)|}, 1 \leq t \leq |A_i| \tag{2}$$

That is, the t-th element of the vector $H_{A_i}(u)$ maintains the percentage of vertices whose attribute value upon the attribute A_i is equal to a_t $(1 \leq t \leq |A_i|)$ w.r.t. all vertices that are l hops away from u. As a result, $H_{A_i}(u)$ maintains the distribution of vertex attribute values of the attribute A_i for the vertices that are near the vertex u. So, the vertex attribute similarity of u and v in terms of their neighbors that are l hops away can be formally defined as:

$$s_l(u, v) = \frac{\sum_{i=1}^{n} \text{sim}(H_{A_i}(u), H_{A_i}(v))}{n} \tag{3}$$

where $\text{sim}(\cdot, \cdot)$ is a similarity function defined on two vectors. In this work, we consider the *cosine* similarity function for computation.

For adjacent vertices u and v, where $(u, v) \in E$, we synthesize the overall vertex attribute similarity, $s(u, v)$, by considering both the vertex attribute similarity of u and v themselves (Eq. (1)) and vertex attribute similarities of all the vertices within the localized vicinity of u and v, respectively, which are up to L hops away from u and v (Eq. (3)):

$$s(u, v) = \sum_{l=0}^{L} \frac{s_l(u, v)}{2^l}. \tag{4}$$

We note that vertices that are l $(1 \leq l \leq L)$ hops away from u and v should contribute less to the vertex attribute similarity between u and v when the value of l grows larger. To account for this, we consider dampening exponentially the vertex attribute similarities of nearby vertices in terms of distances away from u and v, respectively, as presented in Eq. (4). As a result, we assign the vertex attribute similarity $s(u, v)$ as an edge weight $w(u, v)$ of the edge (u, v) in the transformed weighted graph G'. This way, the information of vertex attribute similarity is embedded into G', which will be further explored in order to accommodate structure information in the attributed graph.

Algorithm 1: Vertex attribute embedding (G, L)

```
Input: attributed graph $G(V, E; \mathcal{A})$, maximum neighborhood length $L$
Output: weighted graph $G'(V, E; W)$
1  begin
2      for $e = (u, v) \in E$ do
3          $s(u, v) \leftarrow s_0(u, v)$                                  /* Eq. (1) */
4          for $l = 1$ to $L$ do
5              for $i = 1$ to $n$ do
6                  Construct histograms $H_{A_i}(u)$ and $H_{A_i}(v)$;     /* Eq. (2) */
7              $s_l(u, v) = \frac{\sum_{i=1}^{n} \mathrm{sim}(H_{A_i}(u), H_{A_i}(v))}{n}$   /* Eq. (3) */
8              $s(u, v) \leftarrow s(u, v) + \frac{s_l(u,v)}{2^l}$
                                                                             /* Eq. (4) */
9          $w(e) \leftarrow s(u, v)$
10     return $G'(V, E; W)$
```

Algorithm 1 presents the procedure for vertex attribute embedding. Given an attributed graph G an input, we consider for each edge $e = (u, v)$ a new edge weight $w(e)$ that encodes the vertex attribute similarity, $s(u, v)$, between two vertices u and v. The vertex attribute similarity $s(u, v)$ is first initialized to $s_0(u, v)$ (Line 3). We then expand the neighborhood length, l, of u and v by considering all the vertices that are l hops away from u and v, respectively. For each of different $n = |\mathcal{A}|$ vertex attributes, we construct histograms for neighboring vertices at distance l away from u and v, respectively (Line 6), and the vertex attribute similarity at distance l, $s_l(u, v)$, is computed using a vector-based similarity function $sim(\cdot, \cdot)$, e.g., cosine similarity (Line 7). The vertex attribute similarity $s(u, v)$ is further updated by incorporating $s_l(u, v)$ which is dampened by the distance l (Line 8). The worst-case time complexity of Algorithm 1 is $O(|E| \times L \times |\mathcal{A}| \times d_{\max}^L)$, where $d_{\max}$ is the maximum degree of vertices in the graph G. In practice, we only need to consider the neighborhood L as a very small value ($0 \le L \le 2$), because larger values of L will introduce more vertices that are likely to be in other graph clusters, or with a great percentage of "noisy" vertex attribute values that begin to mismatch with each other.

4.2 *Structure Embedding*

Once the original attributed graph G is transformed to the weighted graph G', the information of vertex attribute similarity is embedded as edge weights of G'. Another aspect of graph clusters yet to be explored is the information of structure closeness. Intuitively, vertices within the same graph cluster are closely connected, while those located in different clusters are typically far apart. We thus take as input the augmented graph G' and further embed the local structure information of vertices for attributed graph clustering.

We consider using a series of short random walks to capture the structure closeness within the localized vicinity of vertices. Random walks have been extensively used as a fundamental data structure to efficiently capture local community structure information of graphs [3, 19, 23, 35]. Specifically, for each vertex $u \in V$, we generate a group of γ truncated random walks rooted from u, denoted as $\mathcal{W}_l^t(u) = (u, v_1, \dots, v_t)$, where $1 \leq l \leq \gamma$, and t denotes the length (i.e., the number of edges) of the truncated random walks, which is often a small value. Each of γ truncated random walks is generated as follows. We start from the vertex $v_0 = u$, and at each step i $(0 \leq i \leq t-1)$, we choose the next vertex $v_{i+1} \in N_1(v_i)$ in the truncated random walk with the following probability:

$$\Pr(v_i, v_{i+1}) = \frac{s(v_i, v_{i+1})}{\sum\limits_{v_j \in N_1(v_i)} s(v_i, v_j)}$$

where $s(v_i, v_{i+1})$ is the weight of the edge (v_i, v_{i+1}) in the graph G', as defined in Eq. (4). That is, the truncated random walks $\mathcal{W}_l^t(u)$ are generated in a *biased* way that the edge (v_i, v_{i+1}) with a higher edge weight will be chosen with a greater probability. Note that edge weights in G' indicate the vertex-wise attribute similarity, as discussed in Sect. 4.1. As a result, the γ truncated random walks generated from u are *attribute-aware* random walks that capture both local structure closeness and vertex attribute homogeneity within the local vicinity of u, if the length t of random walks is set small.

Inspired by the recent advances in language modeling and deep learning [20], we treat each attribute-aware random walk as a short sentence or phrase, and each vertex of the graph as a word in a special language. Our goal is to learn a latent representation $\Phi : v \in V \rightarrow \mathbb{R}^d$ that maps each vertex of the graph into a low-dimensional vector, $\Phi(u)$, while still preserving the clustering properties in Definition 2. Following the intuition of DeepWalk [23], we relax the formulation of random walks in two aspects: (1) a random walk that passes through a vertex $v_i \in V$ as the center of the walk is treated as a bi-directional random walk rooted at v_i. That is, we consider that the random walk originates from v_i and encompasses preceding and following vertices in a window of size $2w$; (2) we ignore the ordering information of vertices in the random walk. The relaxations are particularly useful for the latent representation learning because the order independence assumption well-captures a sense of "closeness" provided by random walks. Furthermore, they greatly simplify the learning process and save a lot of training time. To this end, the latent representation of vertices can be formulated as the following optimization problem:

$$\min_{\Phi}(-\log \Pr(\{v_{i-w}, \dots, v_{i-1}, v_i, v_{i+1}, \dots, v_{i+w}\})) \tag{5}$$

Algorithm 2: Structure embedding (G', w, d, γ, t)

```
Input: weighted graph $G = (V, E; W)$, window size $w$, embedding size $d$, random walks
       per vertex $\gamma$, random walk length $t$
Output: matrix of vertex latent representation $\Phi \in \mathbb{R}^{|V| \times d}$
1 begin
2     for $i = 1$ to $\gamma$ do
3         for $u \in V$ do
4             $\mathcal{W}_i^t(u)$ = RandomWalk($G', u, t$)
5             SkipGram($\Phi, \mathcal{W}_i^t(u), w$)
6     return $\Phi$
```

To solve the optimization problem in Eq. (5), we take advantage of Skip-Gram [20] that maximizes the concurrence probability among the words (vertices) arising within a window w in a sentence (truncated random walk). We further use Hierarchical Softmax [21] and stochastic gradient descent (SGD) to optimize the approximation of probability distributions and parameter estimation.

Algorithm 2 presents the procedure for structure embedding. Given the weighted graph G' as input, we examine every vertex u of G' and embed it into a low-dimensional space as a d-dimensional vector $\Phi(u)$. We generate γ truncated random walks with length t (Line 4). When each truncated random walk originated from u is generated, we use the SkipGram algorithm to update the latent representation in accordance with the objective function in Eq. (5) (Line 5). As the time complexity for the training process of SkipGram is $O(\log|V|)$ for each random walk update, the overall time complexity of Algorithm 2 is $O(\gamma t|E| + |V|\log|V|)$.

5 Attributed Graph Clustering Algorithm

Based on the attributed-aware graph embedding framework discussed in Sect. 4, it becomes straightforward to support clustering on attributed graphs, and the algorithm is sketched in Algorithm 3. Given an attributed graph G, we first embed vertex attribute similarity information into a weighted graph G', where the parameter L regulates the distance of vertex neighborhood to be considered for the quantification of vertex attribute similarity (Line 1). We then embed the structure information of G' by mapping vertices of G' into d-dimensional latent representations, Φ (Line 2), which approximately capture the structure closeness and attribute homogeneity within the local neighborhood of vertices, and thus are important indicators of cluster memberships of vertices. Once the original graph has been transformed into the general latent representations in the d-dimensional space, we can use any traditional data clustering method, such as kMedoids, to partition the d-dimensional vectors representing vertices into k clusters.

Algorithm 3: Attributed graph clustering $(G, k, L, w, d, \gamma, t)$

```
  Input: attributed graph $G$, number of resultant graph clusters $k$, maximum neighborhood
         length $L$, window size $w$, embedding size $d$, random walks per vertex $\gamma$, random
         walk length $t$
  Output: graph clustering $\mathcal{C} = \{C_1, C_2, \ldots, C_k\}$
1 begin
2     $G' \leftarrow$ Vertex Attribute Embedding $(G, L)$
3     $\Phi \leftarrow$ Structure Embedding $(G', w, d, \gamma, t)$
4     $\mathcal{C} \leftarrow k$Medoids$(\Phi, k)$
5     return $\mathcal{C}$
```

It is worth noting that although in Definition 2, we aim to generate *hard* graph clusters, meaning that every vertex can belong to at most one cluster, our proposed attribute-aware graph embedding approach can also support overlapping graph clustering. Once we transform the original attributed graph into its latent representations, Φ, we can apply any hierarchically data clustering method for overlapping graph clusters.

6 Experiments

In this section, we present our experimental studies for the proposed method, AA-Cluster, which is abbreviated for **a**ttribute-**a**ware graph clustering. We compare AA-Cluster with four state-of-the art methods: (1) SA-Cluster [36] that combines vertex attributes and graph structure information through a unified distance measure for attributed graph clustering; (2) BAGC [31] is a Bayesian probabilistic approach for attributed graph clustering; (3) CPIP [18] is based on content propagation with principles of influence propagation as well as random walk (4) DeepWalk [23] that learns the graph structure information as latent features in a low-dimensional space without consideration of vertex attributes. We choose the recommended algorithmic parameters for these, respectively, as mentioned in the corresponding papers. For our method, AA-Cluster, we choose the following default parameter values in the following experimental studies, if not specified otherwise: the vertex neighborhood distance $L = 1$, the number of truncated random walks per vertex $\gamma = 30$, the length of truncated random walks $t = 30$, the window size $w = 30$, and the embedded dimension $d = 40$. All our experiments were carried out in a Linux workstation running RedHat Enterprise Server 6.5 with 16 Intel Xeon 2.3 GHz CPUs and 128 GB of memory.

6.1 Datasets

We examine three real-world attributed graphs and a set of synthetic attributed graphs in our experimental studies. The details of data sets are as follows:

1. **Political Blogs.** This a network of hyperlinks between web blogs on the US politics recorded in 2005.[1] It contains 1490 web blogs as vertices and 19,090 hyperlinks between web blogs as edges. Each blog in the network has an attribute pertaining to its political leaning as either *liberal* or *conservative*;
2. **DBLP.**[2] This is a co-authorship network of computer science authors from four research areas of *database* (including conferences such as SIGMOD, VLDB, PODS, ICDE, and EDBT), *data mining* (including conferences such as KDD, ICDM, SDM, PKDD, and PAKDD), *information retrieval* (including conferences such as SIGIR, CIKM, ECIR, and WWW), and *artificial intelligence* (including conferences such as IJCAI, AAAI, UAI, and NIPS). This network contains 27,199 authors as vertices and 66,832 collaborations as edges. For each author, we consider two categorical attributes: *topic* that is the primary one of 100 research topics extracted from paper titles based on topic modeling [34], and *level* that is determined as follows. If an author published more than 20 papers, the value of level is "*highly prolific*." If the number of published papers is between 10 and 20, the value of level is "*prolific*." If the number of published papers is less than 10, the value of level is "*low prolific*";
3. **Patent.** This is a large patent citation network with vertices representing patents and edges depicting the citations between patents.[3] We extract a subgraph containing all the patents from the year 1988 to 1999. Each patent has six attributes, *grant year*, *number of claims*, *technological category*, *technological subcategory*, *assignee type*, and *main patent class*. There are 1,174,908 vertices and 4,967,216 edges in the network. Note that this is the largest attributed graph in our experimental studies, and most existing attributed graph clustering methods fail to run in this large graph.
4. **Synthetic Graphs.** We generate a series of syntactic attributed, small-world graphs by varying the number of vertices ranging from 1K up to 250K. For vertex attributes, we change the vertex dimensionality, $|\mathcal{A}|$, from 5 to 40 with half of attributes whose values follow the Gaussian distribution, and the remaining half of attributes whose values follow the uniform distribution. The synthetic graphs are primarily used to examine the clustering efficiency and scalability of our proposed method, AA-Cluster.

[1] http://www-personal.umich.edu/~mejn/netdata.

[2] http://dblp.uni-trier.de/xml/.

[3] http://www.nber.org/patents.

6.2 Evaluation Metrics

In order to compare the effectiveness of different attributed graph cluttering methods and assess the quality of resultant graph clusters, we consider the following evaluation metrics in our experimental studies:

1. **Clustering density.** Assume that there are k graph clusters $\mathcal{C} = \{C_1, C_2, \ldots, C_k\}$ generated by some specific attributed graph clustering method. The clustering density is defined as:

$$\mathsf{density} = \sum_{i=1}^{k} \frac{|\{(u, v)|u, v \in V_{C_i}, (u, v) \in E_{C_i}\}|}{|E|} \tag{6}$$

   Density is a quantitative measure indicating the structure closeness of the resultant graph clusters. Empirically, the larger the density value, the better quality of clustering results in terms of structure closeness;
2. **Clustering entropy.** In order to quantify the homogeneity of vertex attribute values in graph clusters, we consider a second evaluation metric, average clustering entropy, which is defined as follows:

$$\mathsf{entropy} = \frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{k} \frac{|V_{C_j}|}{|V|} \text{entropy}(a_i, V_{C_j}) \tag{7}$$

   where

$$\text{entropy}(a_i, V_{C_j}) = -\sum_{l=1}^{|A_l|} p_{ijl} \log p_{ijl}$$

   $n = |\mathcal{A}|$ is the number of vertex attributes in the attributed graph, and p_{ijl} denotes the percentage of vertices in the cluster C_j whose vertex attribute value upon the attribute $A_i = a_{il}$, where $a_{il} \in \text{Dom}(A_i)$. Empirically, the lower the value of entropy, the more homogeneous of vertex attribute values in the resultant graph clusters.

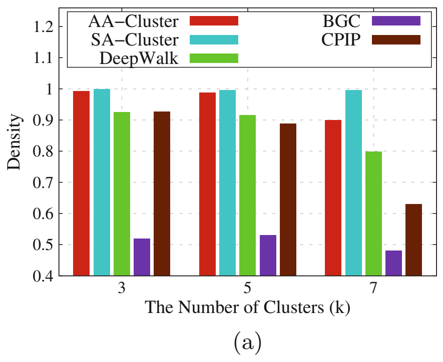
Besides the evaluation metrics for clustering quality, we also examine the runtime cost and scalability of our proposed method, AA-Cluster, as it is important to support clustering real-world large-scale attributed graphs in a fast and potentially scalable way.

6.3 Experimental Results

We report our main experimental results and findings for attributed graph clustering across different datasets. We first perform experimental studies for the clustering quality (in terms of density and entropy) by different attributed graph clustering methods in Sect. 6.3.1. As our method, AA-Cluster, is determined by a series of important parameters, we perform a systematic parametric analysis in Sect. 6.3.2. Finally, we evaluate the scalability of AA-Cluster in synthetic datasets, as reported in Sect. 6.3.3.

6.3.1 Clustering Quality

We first apply different attributed graph clustering methods in the Political Blog graph, and the clustering quality results are illustrated in Fig. 2. By varying the number k of graph clusters, we recognize that the density of graph clusters generated by AA-Cluster is very close to that by SA-Cluster, both of which are consistently higher than the density of BAGC, CPIP, and DeepWalk (Fig. 2a). Meanwhile, the resultant graph clusters by AA-Cluster have a significantly smaller entropy value than BAGC and DeepWalk and close entropy value to CPIP meaning that these graph clusters have more homogeneous vertex attributes (Fig. 2b). For larger k values, CPIP has lower entropy but also lower density value. As a result, AA-Cluster can provide both structurally densely connected and attribute-wise homogeneous graph clusters, whose clustering quality is higher than the graph clusters generated by AA-Cluster and DeepWalk. An interesting observation is that, when $k = 7$, the graph clusters generated by SA-Cluster are structurally imbalanced: There is one large cluster subsuming most vertices of the graph, while the remaining six graph clusters are very small containing a handful of vertices. This results in a high density value but a high entropy value. However, the graph clusters generated by



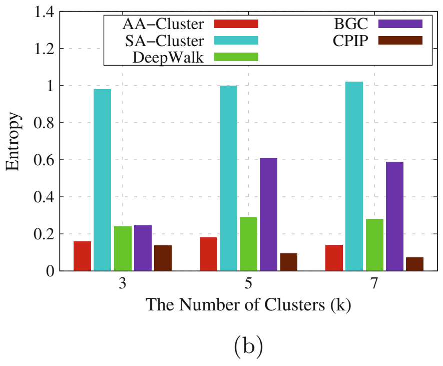


Fig. 2 Clustering quality in political blog dataset. (**a**) Density. (**b**) Entropy

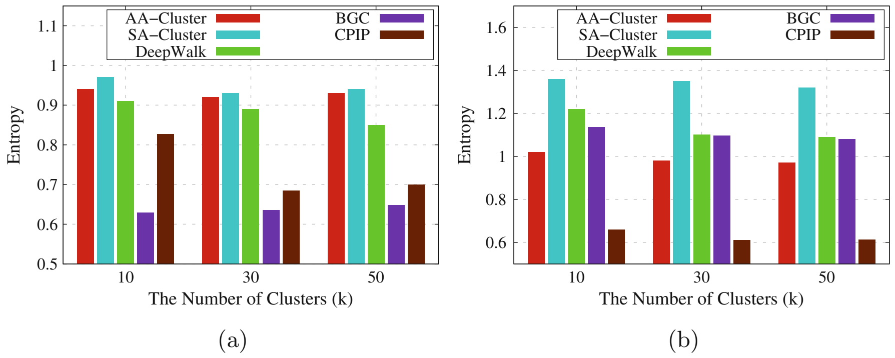


Fig. 3 Clustering quality in DBLP dataset. (**a**) Density. (**b**) Entropy

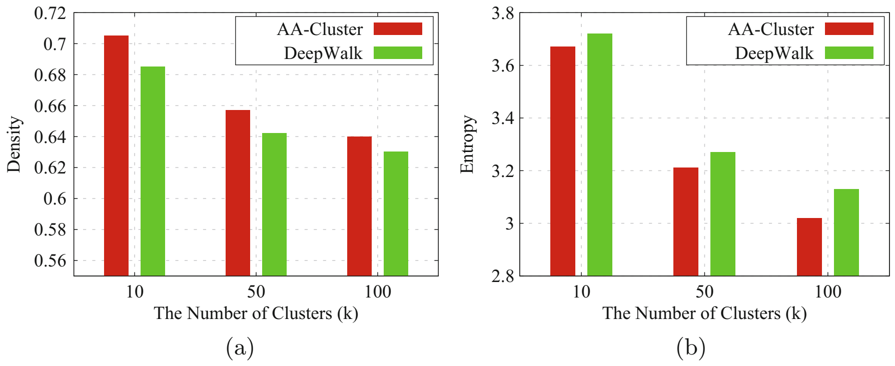


Fig. 4 Clustering quality in patent dataset. (**a**) Density. (**b**) Entropy

AA-Cluster are structurally balanced with the best clustering quality results in terms of both density and entropy.

We then examine different attributed graph clustering methods in the DBLP graph, and the clustering quality results are presented in Fig. 3. By tuning the number k of resultant graph clusters, we can clearly notice that in terms of both density (Fig. 3a) and entropy (Fig. 3b), AA-Cluster outperforms SA-Cluster BAGC and DeepWalk in generating high-quality graph clusters. As similar to previous dataset, CPIP has lower entropy but also lower density value for all k values. This means, while maintaining the attribute homogeneity, it could not keep clusters structurally dense. In addition, the quality results of graph clusters of AA-Cluster are very stable, which are not sensitive to the changes of the number k of graph clusters generated.

We further evaluated different methods in the largest Patent graph. SA-Cluster BAGC and CPIP fail in finishing the clustering computation and returns with runtime memory errors.

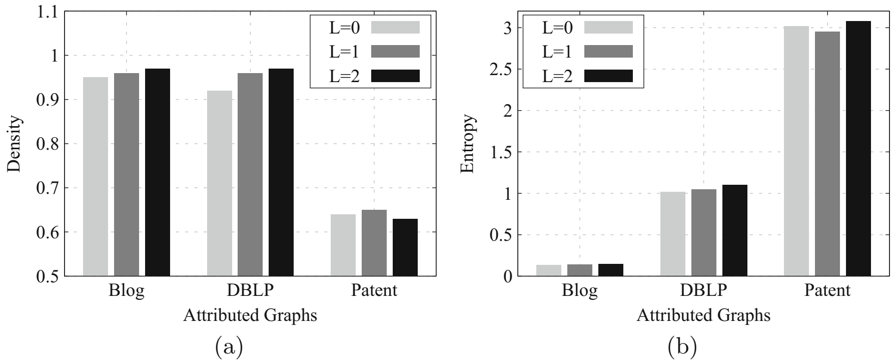


Fig. 5 Clustering quality of AA-Cluster w.r.t. neighborhood distance, L. (**a**) Density. (**b**) Entropy

However, both AA-Cluster and DeepWalk return meaningful graph clustering results, and the clustering quality is reported in Fig. 4. In terms of both density and entropy, we find that AA-Cluster is consistently better than DeepWalk in generating high-quality graph clusters. This indicates that a joint consideration of both graph structure and vertex attribute information will bring better-quality graph clustering results than the one with only graph structure is considered.

6.3.2 Parameter Analysis

It is important to note that our attributed graph clustering method, AA-Cluster, is regulated by a series of important algorithmic parameters. In this section, we will examine how these parameters affect the graph clustering performance of AA-Cluster.

We first study the parameter of neighborhood distance, L, in vertex attribute embedding. The clustering quality results are reported in Fig. 5, in terms of density (Fig. 5a) and entropy (Fig. 5b), respectively. We recognize that by increasing the neighborhood scope, more vertices within the localized region of target vertices are involved for vertex attribute similarity computation. This can be treated as a smoothing step in order to avoid the case that two intracluster vertices happen to share few or even no common vertex attributes. In addition, for the case that two intracluster vertices that share vertex attributes in common but there is no edge connecting them, there are still neighboring vertex attribute similarities that can be leveraged to account for the "closeness" of the intracluster vertices.

As a result, the involvement of neighboring vertices for the quantification of vertex attribute similarity can help improve the graph clustering quality. However, it is not always beneficial to expand the localized neighborhood for vertex attribution value computation. When L is set large, the computational cost of attribute embedding grows as well. More importantly, some noisy vertices with heterogeneous

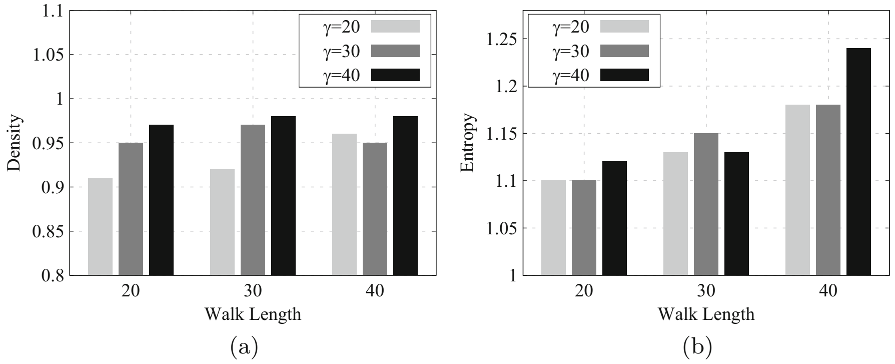


Fig. 6 Clustering quality of AA-Cluster w.r.t. number of walks, γ, in DBLP graph ($k = 10$). (**a**) Density. (**b**) Entropy

vertex attribute values might be involved, thus leading to a drop of entropy. In our experimental studies, we find that $L = 1$ is typically good enough for vertex attribute embedding.

We then examine the parameters pertaining to structure embedding for AA-Cluster. We will report the experimental results in the DBLP network with $k = 10$ graph clusters generated, as we witness very similar trends and findings in the other two graphs.

By varying the length of truncated random walks, t, and the number of truncated random walks rooted per vertex, γ, the graph clustering quality results are illustrated in Fig. 6. We can clearly find that by leveraging more truncated random walks and lengthening the random walks, we can more easily capture the structure closeness of graph clusters. However, the side effect is that we include more vertices with heterogeneous vertex attribute values, thus leading to a decrease in entropy. As a result, there is an intrinsic trade-off when setting the values of t and γ, which are both directly correlated to the clustering quality.

When then test the parameter w of the window size for the SkipGram algorithm used in structure embedding, and the graph clustering quality results are reported in Fig. 7. When w increases, the clustering quality results are enhanced as the density increases and the entropy decreases in the mean time. This suggests that a large window size will benefit the attributed graphs clustering method, AA-Cluster.

6.3.3 Scalability

We also analyze the runtime cost and scalability of the attributed graph clustering method, AA-Cluster, in a series of synthetic graphs. First of all, we create a series of synthetic, small-world graphs with the number of vertices ranging from 1K up to 250K, and the number of vertex attributes, $n = |\mathcal{A}| = 10$, with the values of five vertex attributes following the Gaussian distribution ($\mu = 3$, $\sigma^2 = 5$) and values

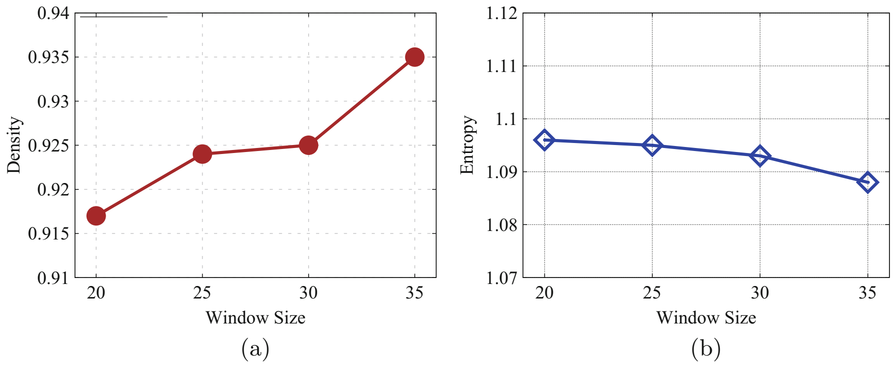


Fig. 7 Clustering quality of AA-Cluster w.r.t. window size, w, in DBLP graph ($k = 10$). (**a**) Density. (**b**) Entropy

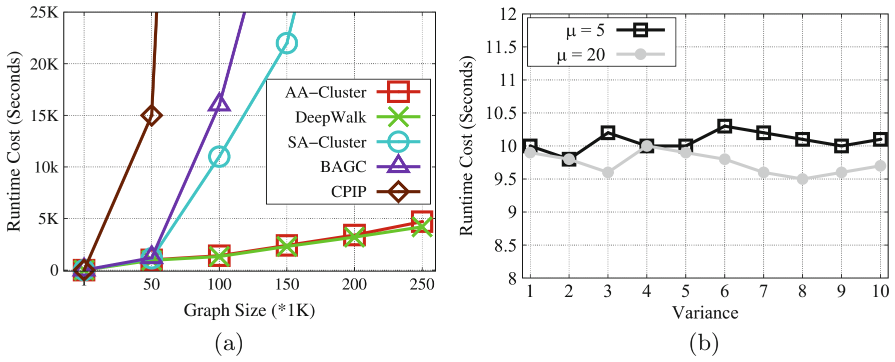


Fig. 8 Runtime cost in synthetic graphs. (**a**) Scalability. (**b**) Variance

of the other five vertex attributes following the uniform distribution. We test all methods on these synthetic graphs, and the runtime results are reported in Fig. 8a.

We find that both AA-Cluster and DeepWalk exhibit excellent scalability for graph clustering, and the runtime cost gap between these two methods is marginal. Note that DeepWalk only deals with graph structure information, while AA-Cluster takes account of both graph structure and vertex attribute information for graph clustering. Therefore, AA-Cluster is both effective and efficient for attributed graph clustering. We also note that SA-Cluster, BAGC, and CPIP cannot scale in large graphs. They give memory error after some points.

We then examine how the value distributions of vertex attributes affect the running time of AA-Cluster. We consider two settings by assigning the mean values of vertex attributes whose values following Gaussian distributions to be 5 and 20, respectively. Meanwhile, we vary the variances, σ^2, from 1 to 20. The runtime results are reported in Fig. 8b. We note that the runtime cost of AA-Cluster is not sensitive to the variances of vertex attribute values. Namely, for the graphs with extremely distributed vertex attribute values, AA-Cluster is still capable of supporting efficient attribute graph clustering upon them.

7 Conclusions

Graph clustering or community detection has played a fundamental role in modeling, structuring, and understanding the large-scale, real-world graphs and networks. In many real-world settings, we are not only concerned with the connectivity structure but also the vertex properties characterized by vertex attributes, for graph clustering. More importantly, we want to study the interplay between graph structure and attribute information in graph clustering with an objective to generating high-quality graph clusters efficiently from real-world, attributed graphs.

In this chapter, we devised a new attributed graph clustering method that combines both vertex attributes and graph structure information within a general, unified attributed-aware graph embedding framework. We design efficient graph embedding algorithms that embed an attributed graph as a low-dimensional latent representation encoding both graph structure closeness and vertex attribute homogeneity. As such, the attribute-aware cluster/community information is well-preserved during the graph embedding. We test our attributed graph clustering method in a series of real-world and synthetic graphs, and the experimental results demonstrate both effectiveness and efficiency of our methods in comparison with state-of-the-art attributed graph clustering techniques.

References

1. Akbas, E., Zhao, P.: Attributed graph clustering: an attribute-aware graph embedding approach. In: Proceedings of the 2017 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2017 (ASONAM'17), pp. 305–308. ACM, New York (2017), http://doi.acm.org/10.1145/3110025.3110092
2. Akoglu, L., Tong, H., Meeder, B., Faloutsos, C.: PICS: parameter-free identification of cohesive subgroups in large attributed graphs. In: Proceedings of the Twelfth SIAM International Conference on Data Mining, Anaheim (SDM'12), pp. 439–450. Society for Industrial and Applied Mathematics, Philadelphia (2012)
3. Andersen, R., Chung, F., Lang, K.: Local graph partitioning using pagerank vectors. In: Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06), pp. 475–486. IEEE, Piscataway (2006)
4. Belkin, M., Niyogi, P.: Laplacian eigenmaps for dimensionality reduction and data representation. Neural Comput. **15**(6), 1373–1396 (2003)
5. Boden, B., Haag, R., Seidl, T.: Detecting and exploring clusters in attributed graphs: a plugin for the gephi platform. In: Proceedings of the 22nd ACM International Conference on Information & Knowledge Management (CIKM'13), pp. 2505–2508. ACM, New York (2013)
6. Bothorel, C., Cruz, J.D., Magnani, M., Micenkova, B.: Clustering attributed graphs: models, measures and methods. Netw. Sci. **3**, 408–444 (2015)
7. Cannataro, M., Guzzi, P.H., Veltri, P.: Protein-to-protein interactions: technologies, databases, and algorithms. ACM Comput. Surv. **43**(1), 1:1–1:36 (2010)

8. Cao, S., Lu, W., Xu, Q.: Grarep: Learning graph representations with global structural information. In: Proceedings of the 24th ACM International on Conference on Information and Knowledge Management (CIKM'15), pp. 891–900. ACM, New York (2015)
9. Dourisboure, Y., Geraci, F., Pellegrini, M.: Extraction and classification of dense communities in the web. In: Proceedings of the 16th International Conference on World Wide Web (WWW'07), pp. 461–470. ACM, New York (2007)
10. Fortunato, S.: Community detection in graphs. Phys. Rep. **486**(3–5), 75–174 (2010)
11. Gong, N.Z., Xu, W., Huang, L., Mittal, P., Stefanov, E., Sekar, V., Song, D.: Evolution of social-attribute networks: measurements, modeling, and implications using Google+. In: Proceedings of the 2012 ACM Conference on Internet Measurement Conference (IMC'12), pp. 131–144. ACM, New York (2012)
12. He, X., Ding, C.H.Q., Zha, H., Simon, H.D.: Automatic topic identification using webpage clustering. In: Proceedings of the 2001 IEEE International Conference on Data Mining (ICDM'01), pp. 195–202. IEEE, Piscataway (2001)
13. Henderson, K., Eliassi-Rad, T., Papadimitriou, S., Faloutsos, C.: HCDF: a hybrid community discovery framework. In: Proceedings of the SIAM International Conference on Data Mining (SDM'10), pp. 754–765. Society for Industrial and Applied Mathematics, Philadelphia (2010)
14. Hu, A.L., Chan, K.C.C.: Utilizing both topological and attribute information for protein complex identification in PPI networks. IEEE/ACM Trans. Comput. Biol. Bioinform. **10**(3), 780–792 (2013)
15. Kim, M., Leskovec, J.: Multiplicative attribute graph model of real-world networks. Internet Math. **8**(1–2), 113–160 (2012)
16. Lattanzi, S., Sivakumar, D.: Affiliation networks. In: Proceedings of the Forty-first Annual ACM Symposium on Theory of Computing (STOC'09), pp. 427–434. ACM, New York (2009)
17. Li, R., Wang, C., Chang, K.C.C.: User profiling in an ego network: co-profiling attributes and relationships. In: Proceedings of the 23rd International Conference on World Wide Web (WWW'14), pp. 819–830. ACM, New York (2014)
18. Liu, L., Xu, L., Wangy, Z., Chen, E.: Community detection based on structure and content: a content propagation perspective. In: 2015 IEEE International Conference on Data Mining, pp. 271–280. IEEE, Piscataway (2015)
19. Macropol, K., Singh, A.: Scalable discovery of best clusters on large graphs. Proc. VLDB Endow. **3**(1–2), 693–702 (2010)
20. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: 27th Annual Conference on Neural Information Processing Systems (NIPS'13), pp. 3111–3119 (2013)
21. Mnih, A., Hinton, G.E.: A scalable hierarchical distributed language model. In: Proceedings of the Twenty-Second Annual Conference on Neural Information Processing Systems (NIPS'08), pp. 1081–1088 (2008)
22. Perozzi, B., Akoglu, L., Iglesias Sánchez, P., Müller, E.: Focused clustering and outlier detection in large attributed graphs. In: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'14), pp. 1346–1355. ACM, New York (2014)
23. Perozzi, B., Al-Rfou, R., Skiena, S.: Deepwalk: online learning of social representations. In: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'14), pp. 701–710. ACM, New York (2014)
24. Roweis, S.T., Saul, L.K.: Nonlinear dimensionality reduction by locally linear embedding. Science **290**(5500), 2323–2326 (2000)
25. Ruan, Y., Fuhry, D., Parthasarathy, S.: Efficient community detection in large networks using content and links. In: Proceedings of the 22nd International Conference on World Wide Web (WWW'13), pp. 1089–1098. ACM, New York (2013)
26. Schaeffer, S.E.: Survey: graph clustering. Comput. Sci. Rev. **1**(1), 27–64 (2007)
27. Steinhaeuser, K., Chawla, N.V.: Identifying and evaluating community structure in complex networks. Pattern Recogn. Lett. **31**(5), 413–421 (2010)

28. Tang, J., Qu, M., Wang, M., Zhang, M., Yan, J., Mei, Q.: Line: large-scale information network embedding. In: Proceedings of the 24th International Conference on World Wide Web (WWW'15), pp. 1067–1077. International World Wide Web Conferences Steering Committee, Geneva (2015)
29. Tenenbaum, J.B., de Silva, V., Langford, J.C.: A global geometric framework for nonlinear dimensionality reduction. Science **290**(5500), 2319–2323 (2000)
30. Xu, Z., Ke, Y., Wang, Y., Cheng, H., Cheng, J.: A model-based approach to attributed graph clustering. In: Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data (SIGMOD'12), pp. 505–516. ACM, New York (2012)
31. Xu, Z., Ke, Y., Wang, Y., Cheng, H., Cheng, J.: GBAGC: a general Bayesian framework for attributed graph clustering. ACM Trans. Knowl. Discov. Data **9**(1), 5:1–5:43 (2014)
32. Yang, T., Jin, R., Chi, Y., Zhu, S.: Combining link and content for community detection: a discriminative approach. In: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'09), pp. 927–936. ACM, New York (2009)
33. Zanghi, H., Volant, S., Ambroise, C.: Clustering based on random graph model embedding vertex features. Pattern Recogn. Lett. **31**(9), 830–836 (2010)
34. Zhai, C., Velivelli, A., Yu, B.: A cross-collection mixture model for comparative text mining. In: Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'04), pp. 743–748. ACM, New York (2004)
35. Zhao, X., Chang, A., Sarma, A.D., Zheng, H., Zhao, B.Y.: On the embeddability of random walk distances. Proc. VLDB Endow. **6**(14), 1690–1701 (2013)
36. Zhou, Y., Cheng, H., Yu, J.X.: Graph clustering based on structural/attribute similarities. Proc. VLDB Endow. **2**(1), 718–729 (2009)
37. Zhou, Y., Cheng, H., Yu, J.X.: Clustering large attributed graphs: an efficient incremental approach. In: Proceedings of the 2010 IEEE International Conference on Data Mining (ICDM'10), pp. 689–698. IEEE, Piscataway (2010)

On Counting Triangles Through Edge Sampling in Large Dynamic Graphs

Guyue Han and Harish Sethu

Abstract Traditional frameworks for dynamic graphs have relied on processing only the stream of edges added into or deleted from an evolving graph, but not any additional related information such as the degrees or neighbor lists of nodes incident to the edges. In this chapter, we propose a new edge sampling framework for big-graph analytics in dynamic graphs which enhances the traditional model by enabling the use of additional related information. To demonstrate the advantages of this framework, we present a new sampling algorithm, called *Edge Sample and Discard* (ESD). It generates an unbiased estimate of the total number of triangles, which can be continuously updated in response to both edge additions and deletions. We provide a comparative analysis of the accuracy and computational complexity of ESD under the new framework against two current state-of-the-art algorithms operating under the traditional framework. The results of the experiments performed on real graphs show that, with the help of the neighborhood information of the sampled edges, the accuracy achieved by our algorithm is substantially better. We also characterize the impact of properties of the graph on the performance of our algorithm by testing on several Barabási–Albert graphs.

1 Introduction

Given the rising significance of social networks in our society, the analysis of their structural properties and the principles guiding their evolution and dynamics have attracted tremendous interest from researchers, sociologists, and marketeers [5, 10, 28]. Social networks can be modeled as graphs with nodes representing users and edges representing the interactions between the users; the study of social networks, therefore, usually translates into a study of extremely large graphs.

G. Han · H. Sethu (✉)
Department of ECE, Drexel University, Philadelphia, PA, USA
e-mail: guyue.han@drexel.edu; sethu@drexel.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_6

In the real world, social networking services (social networking sites or social media), such as Facebook, Twitter, and WeChat, offer prominent examples of fully dynamic graphs. A typical representation of a dynamic graph consists of two components: a connected graph and an edge stream. The stream indicates the addition of a new edge or the deletion of an existing edge from the graph. Social networking service (SNS) providers need to maintain and update their datasets in a real-time fashion. Moreover, service providers may perform various types of analytics on their graph datasets, such as distinguishing different communities, detecting anomalies or spam, and finding the nodes with high betweenness centrality. Real-time graph analytics has the ability to discover important information which can help SNS providers improve existing services, develop new ones, detect anomalous conditions, and respond rapidly to resource management concerns.

One of the key structural properties of interest in social graphs is the triangle, the simplest of graph motifs. The number of triangles is used as one of the signatures of social roles in online discussion groups [33]. The distribution of triangles is a relevant property for spam detection in social networks [4]. The real-time estimation of the number of triangles helps monitor the evolution of the community structure of the graph. The global clustering coefficient can be easily tracked by the changes in the number of triangles and can tell us whether the network is becoming tightly connected, or decentralized [5, 21]. Moreover, a dramatic growth or reduction in the number of triangles in a short time can reflect abnormal behaviors.

In this chapter, we develop a new low-cost sampling algorithm which monitors the edge stream of an evolving graph and is able to, at any instant, provide the current real-time estimate of the number of triangles in it. The goal is for our algorithm to also be adaptable to the case of a static graph.

A dynamic graph, such as one representing a social network, is described by a sequence of edge addition and deletion operations occurring over time. The following tasks may be involved in the management of the graph:

- *Maintenance of the graph datasets.* When a user becomes another user's follower or when a user removes some infrequent contacts from his/her friend list, the system needs to perform edge addition or deletion over the dataset. In addition, the system needs to update the lists of neighbors of the users accordingly.
- *Graph analytics.* Besides the task of maintaining the graph dataset mentioned above, some SNS providers may analyze their social networks quantitatively and qualitatively for better understanding of the networks and improving the existing services.

The fact that a typical dynamic graph-based system needs to maintain the graph dataset as described in the first of the two tasks above whether or not the second task of graph analytics is performed suggests a framework where the minimum available information for graph analytics is all of the information obtained from the first task. Traditional frameworks of dynamic graphs, however, have developed algorithms based only on the sequence of edge additions and deletions, without allowing queries to the graph dataset. In a real scenario, since graph datasets are constantly maintained anyway, it is unnecessary to restrict graph analytics to use

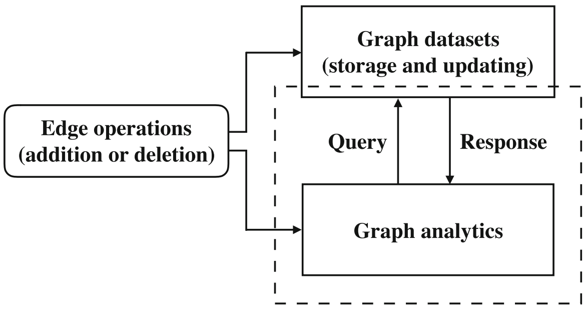


Fig. 1 A framework of graph analytics in a dynamic system

only the information from the edge stream and without use of any information about other graph characteristics related to those edges. Our goal in this chapter is to develop an enhanced framework and demonstrate that algorithms under the new framework can perform substantially better in accuracy and speed compared to those using the traditional and more restrictive edge streaming model.

1.1 A Framework for Graph Analytics

Figure 1 shows the outline of our framework for performing graph analytics in a dynamic system. Each time a new edge operation happens, the system maintains the graph dataset by adding or deleting the edge according to the operation and updates the corresponding information which is determined by the service provided. This comprises the normal functioning of a dynamic graph-based system regardless of whether graph analytics is taken into consideration or not. Note that graph datasets have to be stored somewhere, typically on a server, and can be queried for graph analytics. In almost all real contexts, after all, we would not be counting the triangles in a graph without the graph existing in storage somewhere. Our framework, by assuming the existence of a stored graph dataset and allowing queries on it, achieves a better approximation of the reality of graph-based applications and networks and, as we demonstrate in this chapter, allows the design of algorithms with substantially improved accuracy and speed.

For graph analytics, each new edge operation represents the addition or the deletion of an edge. There are streaming edge algorithms for a variety of analytical purposes, most of which will typically allow only a single pass over the edge stream. Often, in real situations, certain simple queries can be made of the server asking for additional information when processing a new edge. This service providing additional information, via queries, is usually already available on the server since this information is kept updated and maintained by the service providers for offering a number of other necessary services.

In our framework for graph analytics, therefore, we assume the ability to process an edge stream (one pass) and also the ability to query the server maintaining the graph dataset for information on the neighborhood of an edge. The dotted line in the figure frames the real-time graph analytics in a realistic scenario. In this framework, each new edge operation is treated as an arriving edge involving either an addition of an edge or the deletion of an edge. Edges are sampled independently with a certain probability. If an edge is sampled, queries are sent to the server asking for the information on the neighborhood of the two incident nodes of the sampled edge in order to update the estimators used in graph analytics.

We use the above framework to design a new sampling algorithm to keep a running count of the number of triangles in a dynamic graph. The estimation is enabled by the use of neighborhood information. Since the list of neighbors of a user is necessary information for a social network's server to maintain and update anyway, little extra computational/memory costs are expended for querying this additional information.

1.2 Contributions

Based on the new framework/model of a dynamic graph system, we propose a new edge sampling algorithm, called *Edge Sample and Discard* (ESD), which returns an estimate of the total number of triangles in a large dynamic graph by sampling only a tiny fraction of the edges in the graph. For each sampled edge, it samples the presence of triangles at the end points of the edge to update its estimates and then discards the edge (as opposed to holding the edge in memory in a subgraph sample as in [1, 15, 30]). The algorithm works on fully dynamic graphs where both edge deletions and additions are considered, and can be readily applied to static graphs as well. A preliminary version of this work appeared in [12].

In Sect. 2, we introduce the graph model more formally and present the ESD algorithm. We show that the total number of triangles can be estimated from the probabilities with which we sample an edge and one of its neighbor nodes, and whether the sampled edge and the node form a triangle.

Section 3 presents a theoretical analysis of the algorithm and proves that its estimate is an unbiased one. We also derive a bound on the variance of our estimate and draw implications from it. We show that our sampling algorithm for an evolving graph can be readily modified to apply to static graphs.

Section 4 provides a comparative analysis of ESD against two edge sampling algorithms, DOULION [30] and TRIÈST [26] which also can handle both edge additions and deletions, and provide a real-time tracking of the number of triangles. Note that the framework assumed by both DOULION and TRIÈST is different from the one assumed by our algorithm. They are designed assuming the traditional streaming edge framework and do not consider the possibility of using additional information obtained by querying the stored graph dataset on the server. We use several real network graphs with millions of edges to create streams of dynamic graphs. Based on tests on these graphs, we show that, with the use of the neighborhood information of the sampled edges which our new framework allows, the accuracy achieved by ESD is at least one order of magnitude better than the accuracies achieved by DOULION and TRIÈST, while the extra cost of querying for additional information is relatively small. These costs and the associated trade-offs are described in Sect. 4.

Section 4 also evaluates our algorithm on real dynamic graphs. The tests show that our algorithm can generate accurate estimates of the number of triangles in real dynamic graphs. Moreover, we present the influence of the total number of triangles and the clustering coefficient on the accuracy of the estimate of ESD by performing simulations on Barabási–Albert (BA) graphs. The accuracy achieved by our algorithm is better on graphs with a larger number of triangles or a larger global clustering coefficient.

Section 5 concludes the chapter.

1.3 Related Work

Triadic properties such as triangle counts and the global clustering coefficient have been widely studied [7, 32]. Some early works use enumeration or matrix multiplication to compute the exact number of triangles in the graph [8, 9, 18, 24]. Alon et al. [2] propose the theoretically fastest exact triangle counting algorithm, which is based on fast matrix multiplication and runs in $O(|E|^{1.41})$ time. But, it has a high space complexity of $O(|V|^2)$ which renders it largely infeasible for extremely large graphs. Besides, in many cases, the exact answer is not necessary and an approximation is sufficient. Therefore, well-performing approximation methods, which achieve fast runtime and small memory footprint, have attracted tremendous interest.

Eigenvalue-based methods are one class of algorithms used to approximate the global and local number of triangles in the graph [3, 29]. They use the interesting property that the total number of triangles in an undirected graph is $1/6$ of the sum of the cubes of the eigenvalues of its adjacency matrix. But, the computation of matrix multiplication is still very expensive and they only work on static graphs. Hardiman et al. [13] present a method based on a random walk which is capable of estimating both the global and the average clustering coefficient by testing the connectivity of each node in the random walk after the mixing time is reached.

Most of the studies on triangle counting use the graph stream model, where a graph is treated as a stream of edges. Two algorithms for approximating the local number of triangles in both directed and undirected graph are presented in [4]. These two algorithms both require multiple passes over the edge stream and only work on static graphs.

Another class of algorithms uses an edge sparsification approach based on a certain selecting probability to decide whether an edge should be sampled [1, 20, 30]. In [16], a hybrid approach is used which combines edge sparsification with degree-based vertex partitioning. In all of these algorithms, the sample size is not fixed. Algorithms based on reservoir sampling, which use a fixed amount of space for estimating the triadic properties, are described in [15, 25]. However, except for the method presented in [30], none of these algorithms can handle edge deletions in an evolving graph; they work on static graphs and on dynamic graphs with edge additions but not edge deletions.

One approach which works on fully dynamic graphs where both edge additions and deletions are allowed is presented in [17]. It combines the sampling of vertex triples algorithm in [6] and monochromatic sampling method in [22]. The algorithm first performs monochromatic sampling on the original large graph to obtain a sampled graph and then estimates the global clustering coefficient by checking the closure of wedges selected in the sampled graph. The estimate of the total number of wedges in the original graph is obtained by applying the second moment estimation method in [27]. This algorithm has a large memory requirement and cannot provide a real-time update of the estimates. De Stefani et al. [26] propose a method based on reservoir sampling called TRIÈST, which also works on fully dynamic graphs. It adopts random pairing [11], an extension of the reservoir sampling, to solve the problem of accounting for edge deletions. This algorithm uses a fixed sample size and can keep updating the estimates during the processing of the graph. Since the publication of our preliminary work in [12], a similar approach has been used in a recently published report [31]. However, they limit their focus on estimating the number of triangles on static graphs, and do not consider the dynamic case.

As presented in [26], TRIÈST is significantly better than previously known methods in terms of accuracy, space requirement, and the applicability to fully dynamic graphs. However, the framework assumed by TRIÈST does not permit queries to the graph dataset even though the graph dataset in most real applications has to be stored somewhere, such as on a server accessible by API queries. The Edge Sample and Discard (ESD), proposed in this chapter, assumes a different but a more realistic framework allowing access to the graph dataset information which helps substantially improve both the computational/memory costs and the accuracy.

2 The Algorithm

The *Edge Sample and Discard* (ESD) algorithm is designed assuming the framework described in Sect. 1.1. It works on dynamic graphs where both edge deletions and additions are considered. Additional information, the neighboring nodes of the sampled edges, is queried. It is also generalizable to the case of directed graphs, but for clarity of presentation in the paper, we will use undirected graphs in this chapter.

2.1 Preliminaries and Notation

Let $G_t = (V_t, E_t)$ represent an undirected simple graph, where t is the time instant and $t \geq 0$. V_t and E_t are the node set and the edge set at time t, respectively. At the beginning, we have $V_0 = E_0 = \emptyset$.

Consider a stream $\mathcal{S}$ of $((u, v), \beta)$, where (u, v) denotes the edge which is added to or deleted from the graph, and $\beta \in \{+1, -1\}$. $\beta = +1$ indicates that edge (u, v) is added to the graph, and $\beta = -1$ indicates that edge (u, v) is deleted from the graph. For any $t \geq 0$, if a new pair $((u, v), \beta)$ arrives at time t, we update $G_{t-1} = (V_{t-1}, E_{t-1})$ to $G_t = (V_t, E_t)$ with the corresponding edge addition or deletion.

For simplicity, we drop t from the notation and denote by $G = (V, E)$ the most recent update of the graph. Let $\Gamma(v)$ denote the set of neighbors of node $v \in V$, and let $d(v) = |\Gamma(v)|$ denote the node degree of v. A *wedge* is a path of length two, and a *triangle* is a closed wedge (a circular path of length three). Let T denote the set of triangles in G and let $N_T = |T|$.

The goal of this work is to monitor the edge stream of a graph and estimate the value of N_T by examining only a small fraction of the edges and their neighborhoods.

2.2 Edge Sample and Discard

Algorithm 1 presents the pseudo-code of Edge Sample and Discard (ESD) to estimate the total number of triangles given a stream of $((u, v), \beta)$. In our algorithm, we use a global variable T_{est} to record the real-time estimate of the total number of triangles in the current graph. We consider both edge addition and deletion operations; however, as described in Fig. 1 and as in real-life scenarios, the SNS server assumes responsibility for the maintenance and updation of the graph datasets, while the information related to the dataset can be queried and obtained by ESD.

Lines 1–2 in the pseudo-code perform necessary initializations. Lines 3–8 show that for each pair $((u, v), \beta)$ in the stream, we check the value of β and decide whether edge addition or deletion should be performed. Lines 9–13 perform edge sampling and estimate. We use the sampling fraction α as the selecting probability. Each arriving edge has a probability α of being sampled. If an edge is sampled, the *UpdateCount* routine is called. Note that *UpdateCount* works on the graph where the addition or deletion has just been made. Suppose, at time t, an edge $e = (u, v)$ is sampled and *UpdateCount*(u, v, β) is called. Then, we examine the size of the neighborhood of u. For $\beta = -1$, when (u, v) is deleted from G_{t-1}, we check whether node u has neighbors in G_t. For $\beta = +1$, where (u, v) is added to G_{t-1}, we check whether node u has more than one neighbor in G_t. If one of the requirements is fulfilled, we check the value of β and perform the corresponding neighborhood selection. If $\beta = +1$, we pick one node from the neighbor set of u other than v. For example, we select node a from $\Gamma(u) \setminus \{v\}$, and thus the probability of a being

Algorithm 1 The ESD algorithm

```
Require: A graph stream $\mathcal{S}$ and sampling fraction $\alpha$
1: $T_{\text{est}} \leftarrow 0$
2: Create an empty graph $G$
3: for each pair $((u, v), \beta)$ in $\mathcal{S}$ do
4:     if $\beta = +1$ then
5:         Add new edge $(u, v)$ to graph $G$
6:     else
7:         Delete old edge $(u, v)$ from graph $G$
8:     end if
9:     $r \leftarrow$ Random number $\in [0, 1]$
10:    if $r \leq \alpha$ then
11:        UpdateCount$(u, v, \beta)$
12:        UpdateCount$(v, u, \beta)$
13:    end if
14: end for
15: return $T_{\text{est}}$
```

Function used in the ESD algorithm

```
***UpdateCount***$(u, v, \beta)$:
1: if $|\Gamma(u)| > \frac{1+\beta}{2}$ then
2:     if $\beta = +1$ then
3:         Pick random node $a$ uniformly from $\Gamma(u) \setminus \{v\}$
4:         if $a \in \Gamma(v)$ then
5:             $T_{\text{est}} = T_{\text{est}} + \frac{1}{2}\frac{d(u)-1}{\alpha}$
6:         end if
7:     else
8:         Pick random node $a$ uniformly from $\Gamma(u)$
9:         if $a \in \Gamma(v)$ then
10:            $T_{\text{est}} = T_{\text{est}} - \frac{1}{2}\frac{d(u)}{\alpha}$
11:        end if
12:    end if
13: end if
```

picked uniformly at random is $\frac{1}{d(u)-1}$. Since the probability of sampling edge (u, v) is α, the total probability P of selecting (u, v) and then a as a neighbor of u is

$$P = \frac{\alpha}{d(u) - 1}. \tag{1}$$

Given a wedge a-u-v, we check whether the closing edge (a, v) exists by examining $\Gamma(v)$. If $a \in \Gamma(v)$, we update the triangle estimator T_{est}. The estimate is updated by applying Eq. (5) (in Sect. 3). On the other hand, if $\beta = -1$, since edge (u, v) is already deleted from G and v is no longer a neighbor of u, we pick one node from the neighbor set of u. Thus, the total probability P of selecting (u, v) and then one node from the neighborhood of u is

$$P = \frac{\alpha}{d(u)}. \tag{2}$$

After selecting a node from the neighbor set of u, we check whether the selected node is also a neighbor of v. If it is, which means that the subgraph induced by the two incident nodes of the deleted edge (u, v) and the selected node together is a triangle in G_{t-1}, we update the triangle estimator T_{est}.

Given a dynamic graph, ESD avoids using extra space for storing the sample graph by discarding the processed edge and nodes after updating the estimate. The total number of triangles is estimated from the probabilities with which an edge and one of its neighbor nodes are sampled, and whether the sampled edge and the node form a triangle. ESD can provide a real-time estimate of the triangle counts in a dynamic graph as new edges come in or old edges are deleted.

3 Quality of Estimation

In this section, we present the mathematical reasoning behind our triangle estimator and prove that our algorithm provides an unbiased estimate of the total number of triangles with a theoretically tight bound on the variance. We first discuss the case of dynamic graphs allowing only the addition of edges and with edge deletions not considered; we next show that estimating the number of triangles in this additions-only case is not different from that in the case of a fully dynamic graph with both additions and deletions.

Let an ordered tuple (u, v, z) denote the sampled edge (u, v) and the node z selected by *UpdateCount*(u, v, β). The first element in the tuple, u, is one of the incident nodes of the edge and the second element is the other incident node. The third element of the tuple, z, is the node picked from the neighborhood of the first element, u. For example, given a sampled pair $((a, b), +1)$ from the edge stream, we select node c from $\Gamma(a) \setminus \{b\}$ which gives us the ordered tuple (a, b, c).

Let's consider the partially dynamic case with edge additions only. Suppose we have a stream $\mathcal{S}$ of pairs $((u, v), \beta)$ where $\beta = +1$ for each pair in $\mathcal{S}$. If a pair $((u, v), +1)$ arrives at time t, the graph G_{t-1} is updated to G_t as follows:

$$G_t = (V_{t-1} \cup \{u, v\}, E_{t-1} \cup \{(u, v)\}).$$

Let T_t denote the set of triangles in G_t, where $T_0 = \emptyset$. Suppose, at time $t + 1$, we get $(e_{t+1}, +1)$ from the stream, where $e_{t+1} = (u, v)$ is an edge arriving at time $t + 1$, so we have an updated graph G_{t+1}. Let $H(e_t, G_t)$ denote the set of triangles composed of edge e_t in graph G_t. We can obtain that $T_{t+1} = T_t \cup H(e_{t+1}, G_{t+1})$. Since $T_t \cap H(e_{t+1}, G_{t+1}) = \emptyset$,

$$|T_{t+1}| = |T_t| + |H(e_{t+1}, G_{t+1})|. \tag{3}$$

According to Eq. (3), we can obtain

$$|T_t| = \sum_{i=0}^{t} |H(e_i, G_i)|. \tag{4}$$

Let $\mathcal{Q}_t$ denote the set of all ordered tuples (u, v, z) that have a nonzero probability to be observed when processing a new arriving pair (e_t, β) from stream $\mathcal{S}$. Let $\mathcal{T}_t \subseteq \mathcal{Q}_t$ be the set of all ordered tuples in $\mathcal{Q}_t$ of which the three elements form a triangle in G_t. Note that (u, v, z) and (v, u, z) are two different tuples but the three elements in each of them induce the same triangle in the graph.

Suppose (e_t, β) is sampled, and then one node is picked from the neighborhood of each incident node of e_t. So, the same triangle may be observed twice during the sampling period. Thus, we can obtain

$$|\mathcal{T}_t| = 2|H(e_t, G_t)|.$$

Consider $\mathcal{T}_t' \subseteq \mathcal{T}_t$ as the set of ordered tuples obtained by sampling $(e_t, +1)$, where the three elements of each ordered tuple in $\mathcal{T}_t'$ form a triangle in G_t. Let $P(r)$ be the probability that an ordered tuple $r = (u, v, z) \in \mathcal{T}_t'$ is sampled. By adopting the Horvitz–Thompson construction[14], we come up with the linear estimator:

$$H^t_{\text{est}} = \omega \sum_{r \in \mathcal{T}_t'} \frac{1}{P(r)}. \tag{5}$$

where ω is a weight parameter, and $\omega = |H(e_t, G_t)|/|\mathcal{T}_t| = 1/2$.

Let $\Delta_k = (u, v, z)$ be an element in $\mathcal{T}_t$, where $k \in [1, |\mathcal{T}_t|]$. Remember that the subgraph induced by the three elements of Δ_k in G_t is a triangle. Let δ_k denote the existence of Δ_k in the set $\mathcal{T}_t'$. We have

$$\delta_k = \begin{cases} 1 & \Delta_k \in \mathcal{T}_t' \\ 0 & \Delta_k \notin \mathcal{T}_t' \end{cases}$$

Taking the expectation of H^t_{est},

$$\begin{aligned} \mathrm{E}\left[H^t_{\text{est}}\right] &= \omega \sum_{k=1}^{|\mathcal{T}_t|} \mathrm{E}\left[\delta_k \cdot \frac{1}{P(\Delta_k)}\right] \\ &= \omega \sum_{k=1}^{|\mathcal{T}_t|} P(\Delta_k) \cdot \frac{1}{P(\Delta_k)} \\ &= |H(e_t, G_t)| \end{aligned}$$

The expected number of triangles composed of edge e_t obtained by our estimator is equal to the actual number of triangles composed of edge e_t in G_t. Applying Eq. (4) and Eq. (5), we get

$$|T_t|_{\text{est}} = \omega \sum_{i=0}^{t} \sum_{r \in \mathcal{T}_t'} \frac{1}{P(r)}$$

According to the linearity of the expectation, $\mathrm{E}[|T_t|_{\mathrm{est}}] = |T_t|$. So, we have an unbiased estimator to approximate the total number of triangles in G_t. The variance of H^t_{est} is

$$\begin{aligned}\mathrm{Var}[H^t_{\mathrm{est}}] &= \mathrm{E}[(H^t_{\mathrm{est}} - \mathrm{E}[H^t_{\mathrm{est}}])^2] \\ &= \mathrm{E}\left[{H^t_{\mathrm{est}}}^2\right] - |H(e_t, G_t)|^2\end{aligned}$$

Expanding $\mathrm{E}\left[{H^t_{\mathrm{est}}}^2\right]$, we get

$$\mathrm{E}\left[{H^t_{\mathrm{est}}}^2\right] = \omega^2\,\mathrm{E}\left[\sum_{i=1}^{|\mathcal{T}_t|}\left(\frac{\delta_i}{P(\Delta_i)}\right)^2 + \sum_{i\neq j}^{|\mathcal{T}_t|}\frac{\delta_i\delta_j}{P(\Delta_i)P(\Delta_j)}\right]$$

Suppose the sampling fraction is α, and the maximum degree in G_t (the graph upon the most recent update) is d_{max}, then we have $P(\Delta_i) \geq \frac{\alpha}{d_{\mathrm{max}}-1}$ for any $i \in [1, |\mathcal{T}_t|]$. Letting $m = \frac{\alpha}{d_{\mathrm{max}}-1}$, we can obtain

$$\mathrm{E}\left[\sum_{i=1}^{|\mathcal{T}_t|}\left(\frac{\delta_i}{P(\Delta_i)}\right)^2\right] \leq |\mathcal{T}_t|\frac{1}{m}$$

According to the definition of $\mathcal{T}_t$, each ordered tuple in $\mathcal{T}_t$ represents the triangle which is part of edge e_t in G_t. So, the two triangles represented by any two ordered tuples Δ_i and Δ_j in set $\mathcal{T}_t$, where $i \neq j$, must have edge e_t as a shared edge. Thus,

$$\mathrm{E}\left[\sum_{i\neq j}^{|\mathcal{T}_t|}\frac{\delta_i\delta_j}{P(\Delta_i)P(\Delta_j)}\right] = \frac{2}{\alpha}\left(\frac{|\mathcal{T}_t|}{2}\right)^2$$

So, the variance of our estimator is bounded as follows:

$$\mathrm{Var}[H^t_{\mathrm{est}}] \leq \frac{|\mathcal{T}_t|(d_{\mathrm{max}}-1)}{4\alpha} + \left(\frac{1}{2\alpha} - 1\right)\left(\frac{|\mathcal{T}_t|}{2}\right)^2$$

Since $\mathrm{Var}[|T_t|_{\mathrm{est}}] = \sum_{i=0}^{t}\mathrm{Var}[H^t_{\mathrm{est}}]$, we have

$$\mathrm{Var}[|T_t|_{\mathrm{est}}] \leq \frac{|T_t|(d_{\mathrm{max}}-1)}{2\alpha} + \sum_{i=0}^{t}|H(e_t, G_t)|^2\left(\frac{1}{2\alpha} - 1\right)$$

For any $t > 0$, when the degree of the incident nodes of the sampled edge in G_t, where t is the time step that the edge is sampled, are all equal, the equality in the above bound holds. Therefore, the bound derived above on the variance of our estimate, $\mathrm{Var}[|T_t|_{\mathrm{est}}]$, is a strict upper bound.

Further, applying Chebyshev's inequality:

$$P(||T_t|_{\text{est}} - |T_t|| \geqslant \epsilon|T_t|) \leqslant \frac{\text{Var}[|T_t|_{\text{est}}]}{\epsilon^2|T_t|^2} \tag{6}$$

The above shows that the relative error of the estimate is influenced by the sampling fraction and the properties of the graph. The error is increased as the value of the sampling fraction is decreased and the estimate achieves a better accuracy on a graph with more triangles. Moreover, the local clustering coefficient also affects the estimate; the algorithm can achieve a better accuracy on a graph where most of the nodes have a higher clustering coefficient. In general, ESD achieves a better estimate on graphs with a higher global clustering coefficient.

The proof for the fully dynamic case with edge deletions is similar to the case with additions described above. Suppose we keep performing edge addition up to time t. Thus, at time t, we have a graph $G_t = (V_t, E_t)$ and T_t, the set of triangles in G_t. Suppose, at time $t+1$, we get $(e_{t+1}, -1)$ from the stream, indicating an edge deletion, so we have an updated graph G_{t+1}. We can easily obtain that $T_{t+1} = T_t \setminus H(e_{t+1}, G_t)$. In other words, we have

$$|T_{t+1}| = |T_t| - |H(e_{t+1}, G_t)|.$$

As proved before, we have an unbiased estimator H^t_{est} for estimating $|H(e_t, G_t)|$, so in the edge deletion case, we use the same estimator to estimate the decreased number of triangles caused by deleting e_{t+1}.

3.1 Static Graphs

Although ESD is designed for implementation on dynamic graphs, it can be easily extended to work on static graphs. In the dynamic case, a triangle can be detected only when the new coming edge is the closing edge of a wedge already in the graph; however, in static graphs, each of the three edges of a triangle appears in the edge stream, so one triangle can be detected every time the new coming edge is part of this triangle. Thus, for static graphs, the value of the weight parameter ω of the estimator is one-third of the one in the dynamic case.

Let $\mathcal{S}'$ be the set of ordered tuples which represent the triangles observed in the sampling period by an edge stream which delivers random edges from the static graph. Let $P(s)$ be the probability that a tuple $s \in \mathcal{S}'$ is sampled. By applying the Horvitz–Thompson construction, we have

$$|T|_{\text{est}} = \omega \sum_{s \in \mathcal{S}'} \frac{1}{P(s)}$$

where the weight parameter is $\omega = 1/6$.

4 Performance Analysis

In this section, we perform a comparative analysis of the performance of ESD against TRIÈST [26] and DOULION[30]. Both TRIÈST and DOULION, like ESD, can provide a real-time estimate of the total number of triangles by performing edge sampling on an edge stream of a fully dynamic graph. However, TRIÈST and DOULION assume a traditional streaming graph framework, where a graph can be processed only via a stream of edges. As our comparative analysis will show, the use of additional information by ESD already available and kept updated for graph maintenance as per our framework substantially improves the accuracy.

We use real, simple, and undirected graphs from the Network Repository site [23] to create streams of addition-only graphs and fully dynamic graphs. Table 1 summarizes some vital properties of these graphs. We also evaluate our algorithm on two real dynamic graphs from [19] and [34].

All of the tests were run on an iMac with 16 GB 1600 MHz DDR3 memory and 2.7 GHz Intel Core i5 processor. Our primary goal in this project was to democratize Big Data analysis and make it feasible even on ordinary desktops. It is for this reason that we deliberately choose ordinary computers and we want to show the capability as well as the limits of sampling algorithms. As a result, the datasets considered in this chapter are not very large (e.g., billions of nodes), but the point we are making is not that we can use our algorithms on large datasets but that our sampling strategy is a highly efficient one that substantially increases what one can accomplish with an ordinary computer.

4.1 Complexity

Fast runtime and small space requirement are two vital goals of a good sampling algorithm.

Table 1 Properties of the graphs used in the experiments

Graph	$\|E\|$	$\|V\|$	N_T	η
socfb-UCLA	7.47e+05	2.05e+04	5.11e+06	0.1431
socfb-Wisconsin	8.35e+05	2.38e+04	4.86e+06	0.1201
com-Amazon	9.26e+05	5.49e+05	6.67e+05	0.2052
com-DBLP	1.05e+06	4.26e+05	2.22e+06	0.3064
web-Stanford	1.99e+06	2.82e+05	1.13e+07	0.0086
web-Google	4.32e+06	8.76e+05	1.34e+07	0.0552

$|E|$ is the number of edges, $|V|$ is the number of nodes, N_T is the number of triangles, and η is the global clustering coefficient

4.1.1 Runtime

Consider the partially dynamic case where edge additions happen $|E|$ times and there are no edge deletions. Suppose the neighbors of each node in a graph are stored in a sorted list. This allows a determination of whether a node is a neighbor of another specific node in $O(\log d)$ steps where d is the degree of that specific node.

In TRIÈST, an edge reservoir is maintained and updated. Suppose the size of the edge reservoir is M. At time t ($t > M$), the probability of updating the edge reservoir is M/t, so the expected number of times that the edge reservoir is updated is $M + \sum_{t=M+1}^{|E|} \frac{M}{t} \approx M + M \ln |E|$. Each time the edge reservoir is updated, TRIÈST checks the number of triangles composed of the newly sampled edge in the sample graph. Suppose d_s is the maximum degree in the sample graph, the computational complexity of TRIÈST is $O(Md_s \log |E| \log d_s)$.

As presented in [30], the computational complexity of DOULION is $O(p|E| + (p|E|)^{\frac{2\omega}{\omega+1}})$, where ω is 2.371 and p is the probability of sampling an edge.

In the case of ESD, suppose $p|E|$ is the number of edges which are sampled, and each query of the neighbor nodes takes O(1). Then, for each sampled edge, we take a maximum of $O(\log d_G)$ steps to sample a neighboring node and determine if the node and the sampled edge form a triangle. The complexity of ESD, therefore, is $O(p|E| \log d_G)$. Besides, the server needs to take $O(p|E|)$ to respond to the queries.

Table 2 summarizes the complexity of the three algorithms for the case with no edge deletions. ESD is faster than DOULION when $\log d_G < (p|E|)^{0.41}$ (the number of edges sampled is not too small). In fact, on all real graphs today and reasonable sample sizes, ESD enjoys a lower computational complexity than DOULION. For ESD and TRIÈST, since $\log |E| > \log d_G$ for large graphs, ESD is faster than TRIÈST when sampling the same number of edges ($p|E| = M$).

Moreover, for each edge deletion, both DOULION and TRIÈST have to check whether the deleted edge is in the sample set or not, and update the estimates. While in ESD, it avoids looking up the sample set and processes edge deletions with a sampling probability. So, our algorithm is substantially faster than DOULION and TRIÈST, especially when facing a large amount of edge deletions.

Table 2 The complexity of the three algorithms

Algorithm	Time complexity	Server-side time complexity	Space complexity	Server-side space complexity
ESD	$O(p\|E\| \log d_G)$	$O(p\|E\|)$	$O(d_G)$	$O(d_G)$
DOULION[a]	$O(p\|E\| + (p\|E\|)^{\frac{2\omega}{\omega+1}})$	N/A	$O({V_s}^2)$	N/A
TRIÈST	$O(Md_s \log \|E\| \log d_s)$	N/A	$O(M)$	N/A

[a]The algorithm given in [30] cannot provide a real-time estimate; it only updates the estimate once after processing the entire graph

Note that ESD uses a different framework from DOULION and TRIÈST, neither of which involve querying the server for additional information. Given the framework used by ESD, it involves an additional server-side cost in responding to the queries. As shown in Table 2, ESD achieves a substantially better trade-off saving computational and memory costs with a small amount of extra effort on the part of the server.

4.1.2 Space

At first sight, it may appear as though the framework used in this chapter to develop the ESD algorithm requires the storage of the entire graph, while DOULION and TRIÈST only have to store the sampled graph. However, this mischaracterizes the actual storage needs under these frameworks. In real contexts, even streaming graph data for a fully dynamic graph are ultimately generated by a system/server which maintains and keeps updated a graph dataset. After all, a well-maintained graph dataset is essential for the normal functioning of most applications relying on the graph. Before, after, and in the midst of any graph analytics, in most real contexts, the graph datasets are still kept stored somewhere. So even for the traditional streaming graph model (used by TRIÈST and DOULION), the existence and the storage needs of the complete graph dataset *cannot* and *should not* be ignored. It is unrealistic to assume that, after performing a streaming graph algorithm, one would only store the sampled graph and not store anywhere the large dynamic graph observed so far. Therefore, the cost of storing the dynamic graph is a necessity for all of the three algorithms and their respective frameworks.

For the graph analytics part, our algorithm avoids the requirement of extra memory to store the sampled edges by performing independent collections of edges and discarding edges after updating the estimators. The list of the neighbor nodes of the sampled edge are required for estimating which leads to a memory requirement $O(d_G)$ where d_G is the maximum degree in the original graph. As described in Sect. 1.1, the graph datasets are maintained regardless of whether the graph analytics is applied or not. Thus, for the server-side space complexity, we do not include the cost for maintaining the graph datasets, and only count the extra space complexity required by our algorithm.

DOULION uses a certain probability p to sample edges in the stream, and the samples are maintained in the memory during the entire process. So, the amount of memory used is not fixed and partially depends on the algorithm used to calculate the exact number of triangles in the sampled graph. In [30], the fast matrix multiplication is used to count triangles in the sampled graph, so the space complexity is $O({V_s}^2)$, where V_s is the number of nodes in the sampled graph. TRIÈST is a reservoir sampling-based algorithm which uses a fixed amount of memory to store the sampled edges.

4.2 Partially Dynamic Case

We show the comparison of the performances of ESD with TRIÈST and DOULION on dynamic graphs where only edge additions are considered. The edge stream is generated by permuting the edges uniformly at random.

We consider the relative error in estimating the triangle number as a measure of the accuracy. The relative error is measured as:

$$\text{Relative error} = \frac{\text{Average estimate} - \text{Actual value}}{\text{Actual value}},$$

where the average estimate is the mean of the estimated value over 100 independent runs.

Table 3 shows relative errors in estimating the total number of triangles for each of the three algorithms. We sample 1% of the edges for each graph. As shown in the table, ESD achieves better accuracy than the other algorithms on all the graphs. On most of the graphs, the relative errors obtained by ESD are at least one order of magnitude smaller than the errors obtained by DOULION and TRIÈST.

To further compare the accuracy of the three algorithms, we use the normalized root mean square error:

$$\text{NRMSE} = \frac{\sqrt{\text{E}[(\text{estimate} - \text{Actual value})^2]}}{\text{Actual value}},$$

Figure 2 depicts the average NRMSEs based on 100 independent runs for each graph as the sample sizes are increased. We can see from the figure that ESD has the smallest NRMSEs in all cases. Especially when the sample size is small, the NRMSEs of ESD are almost one order of magnitude smaller than the NRMSEs of DOULION and TRIÈST. On most of the graphs, our algorithm achieves similar NRMSEs to the other two algorithms with one-ninth of the sample size used by these two algorithms. In other words, to achieve equivalent accuracy, ESD requires fewer samples and thus reduces the computational cost.

Table 3 The relative errors in the estimates of the total number of triangles

		Triangles N_T		
		Relative error (%)		
Graph name	Sample size	ESD	DOULION	TRIÈST
socfb-UCLA	7,476	0.0958	1.6274	3.5433
socfb-Wisconsin	8,359	0.5380	4.2736	2.6817
com-Amazon	9,258	0.2329	6.4262	3.4856
com-DBLP	10,498	0.4301	1.5458	3.4481
web-Stanford	19,926	0.9796	3.1822	5.6751
web-Google	43,220	0.0336	2.2997	3.2821

Sample size is the number of edges sampled

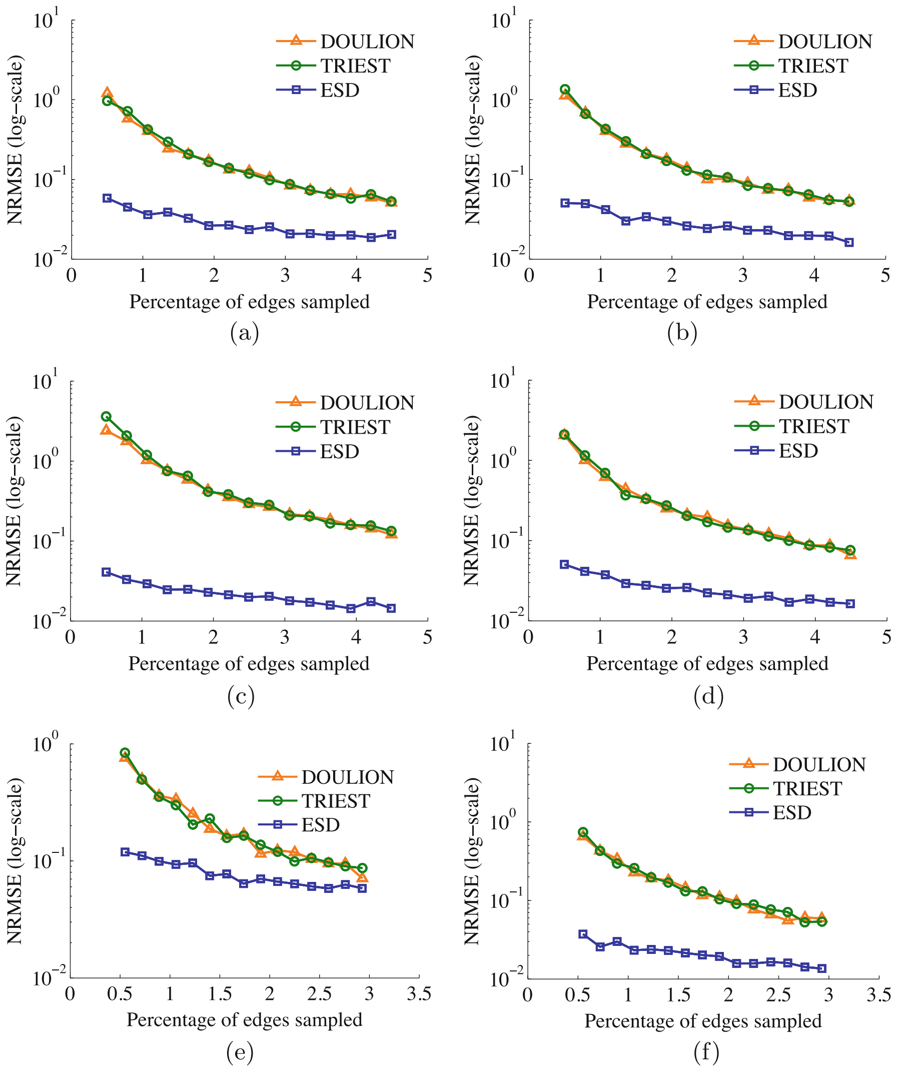


Fig. 2 Comparison of NRMSEs of the estimates for three algorithms over 100 independent runs. **(a)** socfb-UCLA, **(b)** socfb-Wisconsin, **(c)** com-Amazon, **(d)** com-DBLP, **(e)** web-Stanford, **(f)** web-Google

4.3 Fully Dynamic Graphs

In the experiments for the fully dynamic case, we use the model presented in [26] to simulate the deletions or additions of nodes or edges.

We first tested the performances of the three algorithms on dynamic graphs where both edge additions and deletions are considered. For each test on the graph, we first

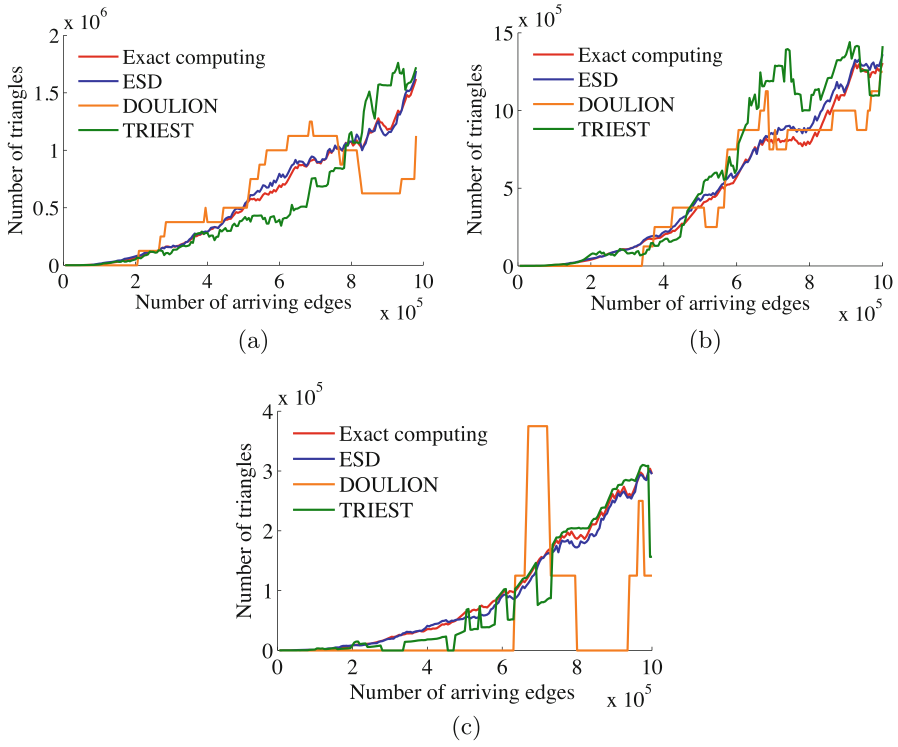


Fig. 3 Comparison of the estimated values of total number of triangles in dynamic case. **(a)** socfb-UCLA, **(b)** socfb-Wisconsin87, **(c)** com-DBLP

generate a stream of edges by randomly permuting the edges. Initially, an empty graph G is created. The arrival of each edge in the stream is treated as an edge addition, and each new edge is added into G. A probability $p_e = 0.0001$ is used to decide whether a deletion event should be performed after each edge addition is made. If a deletion event happens, every edge in G has a probability $p_d = 0.01$ of being deleted.

Figure 3 shows the comparison of the estimates of the triangle number for three algorithms. The red line indicates the actual number of triangles obtained by the exact triangle computing algorithm. For all graphs, the final sample sizes of both DOULION and ESD are approximately equal to 10,000, while the sample size of TRIÈST is fixed at 10,000.

As shown in the figure, ESD has the best performance among the three in tracking the changes in the number of triangles, even at small sample sizes. In the case of DOULION, however, the edge deletion affects the accuracy of its estimate. If many deleted edges are edges held in the sample set, the sample set shrinks quickly, significantly reducing the accuracy of estimates made by DOULION. For example, in the com-DBLP graph, the estimate obtained by DOULION is sometimes more than twice as large as the actual value.

TRIÈST uses reservoir sampling with a fixed size of the sample set. If the deleted edge is an edge in the sample set, it would be removed from the sample set, and the edge deletion in the sample set would be compensated by future edge insertion. So, TRIÈST can maintain a sample set with fixed number of edges during the entire sampling period. In other words, the number of edges sampled by TRIÈST is always larger than the number of edges sampled by ESD and DOULION in the simulations. The figure shows that ESD achieves a better performance than TRIÈST in terms of the accuracy even though TRIÈST samples more edges than ESD.

Besides edge deletion, in the real world, node deletion also occurs. Deletion of a single node can be modeled as a sequence of deletions of edges adjacent to that node. We also tested the performances of the three algorithms on dynamic graphs with node deletions. For each test on the graph, we use a probability $p_e = 0.0001$ to decide whether a deletion event should be performed after the occurrence of each edge addition. If a deletion event happens, every node in the current graph has a probability $p_d = 0.001$ of being deleted.

Figure 4 plots the comparison of the estimates of the number of triangles for the three algorithms. For all graphs, the sampling probability is set to 0.02 for DOULION and is set to 0.01 for ESD, while the sample size of TRIÈST is fixed at 10,000.

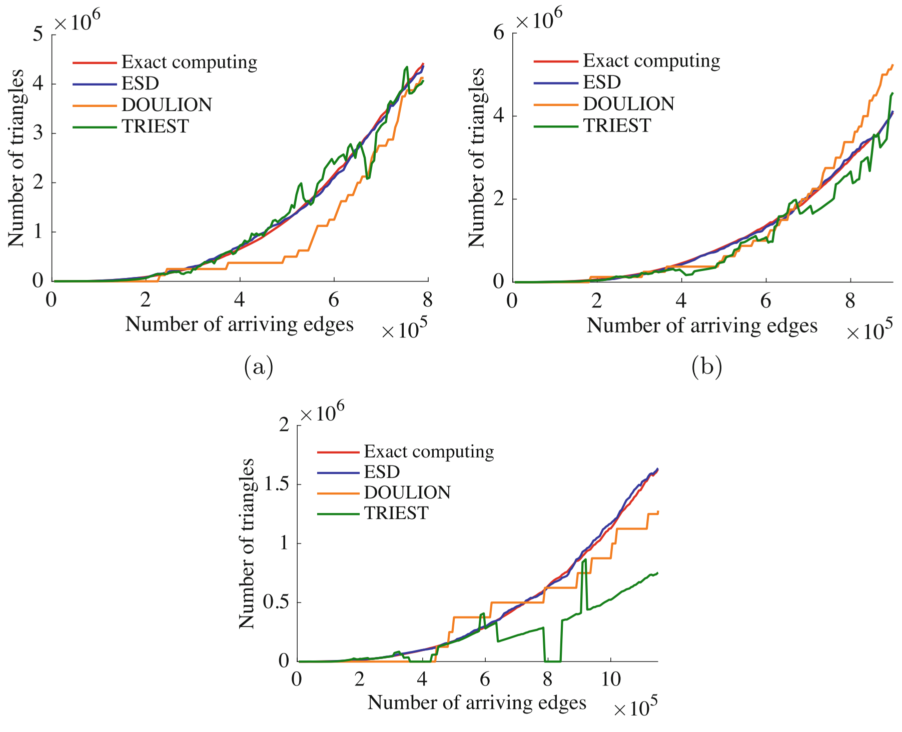


Fig. 4 Comparison of the estimated values of total number of triangles in dynamic case. **(a)** socfb-UCLA, **(b)** socfb-Wisconsin87, **(c)** com-DBLP

As plotted in the figure, ESD achieves the best performance among the three in estimating the number of triangles in dynamic graphs with node deletions. The red line which indicates the actual number of triangles is extremely close to the blue line which plots the estimates obtained by ESD. The closeness between the red line and the blue line indicates that our algorithm is capable of providing accurate estimates of number of triangles in a real-time fashion. Moreover, for all graphs, ESD has the smallest final sample size. In other words, our algorithm samples fewer edges, but achieves higher accuracy than the other two algorithms in tracking the number of triangles.

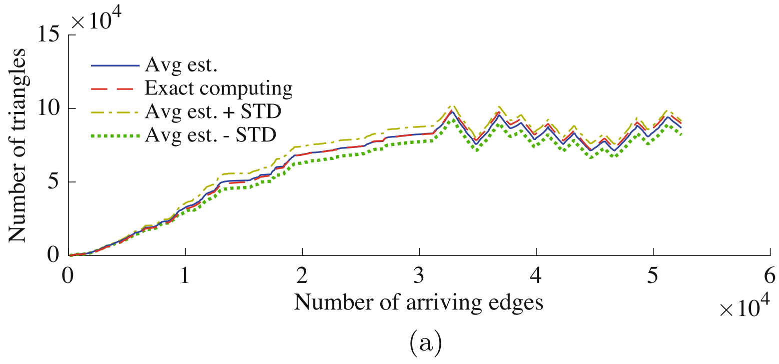
Besides the abovementioned model, we also tested the performance of our algorithm on two real dynamic graphs. Oregon-2 dataset [19] contains 9 autonomous system (AS) graphs which represent AS peering information inferred from Oregon route views. It was collected from March 31, 2001 to May 26, 2001. Yahoo! Message dataset [34] which has 28 graphs was generated by a small subset of Yahoo! Messenger users from different zip codes for 28 days starting from April 1, 2008. Graphs in each of the datasets are timestamped. Each of the graphs is processed sequentially according to its time order. Edges which are not present in the previous graph, but are in the current graph, are treated as edge additions, and edges which are present in the previous graph, but are not in the current graph, are treated as edge deletions. Thus, both of the two datasets exhibit the addition and deletion of the edges over time.

Figure 5 shows the estimation of the total number of triangles on real dynamic graphs when the sampling fraction is $\alpha = 0.01$. As shown in the figure, ESD has a good performance on real graphs in terms of the accuracy and the variance. Even on the Yahoo! Message graph, where the number of triangles changes frequently and dramatically, our algorithm is still capable of tracking variation on the number of triangles accurately.

4.4 Relationship to Graph Properties

We show the influence of the properties of the graph on the performance of ESD by testing on several Barabási–Albert (BA) graphs. For ease in illustrating this relationship, we only consider edge additions in this set of experiments. We ran ESD on BA graphs with the same number of nodes, but with different numbers of edges and different powers of the preferential attachment. For each BA graph, we start with an Erdös–Rénzi graph with 100 nodes. Then, in each time step, one node is added to the graph, and the new node initiates dozens of edges to old nodes. The probability that an old node is selected is given by:

$$P(i) \sim {d_i}^{\gamma}$$



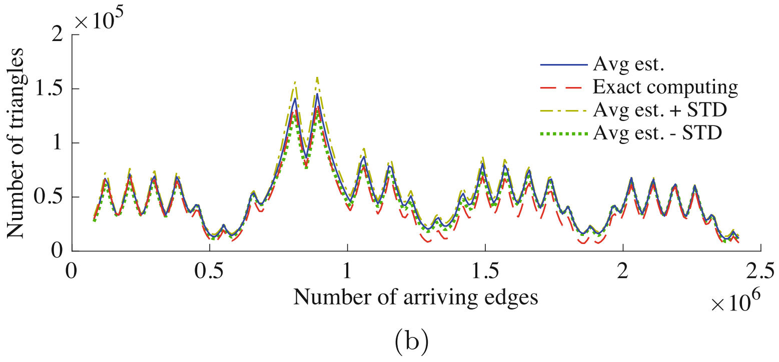


Fig. 5 Estimation of the total number of triangles in dynamic case. **(a)** Oregon-2, **(b)** Yahoo! Messenger

Table 4 Properties of the Barabási–Albert (BA) graphs used in the experiments

Graph	γ	N_T	η	$\|E\|$	$\|V\|$
BA-1	1.5	281,296	0.00255	200,500	20,000
BA-2	1.5	1,046,132	0.00381	399,500	20,000
BA-3	1.5	6,874,145	0.01195	996,500	20,000
BA-4	1.0	3,526,361	0.02615	1,474,100	20,000
BA-5	1.5	3,473,097	0.00818	757,700	20,000
BA-6	2.0	3,464,420	0.00233	598,500	20,000

γ is the power of the preferential attachment, N_T is the number of triangles, and η is the global clustering coefficient

where d_i is the degree of node i in the current time step and γ is the power of the preferential attachment. Table 4 lists some basic properties of the BA graphs used in these simulation experiments.

Figure 6 shows the ratio of the average estimated total number of triangles to the actual value for each BA graph with increasing number of edges sampled. The error bars represent the 95% confidence intervals. The red line indicates 1, when the estimated and the actual values are equal. For all of the graphs, the same

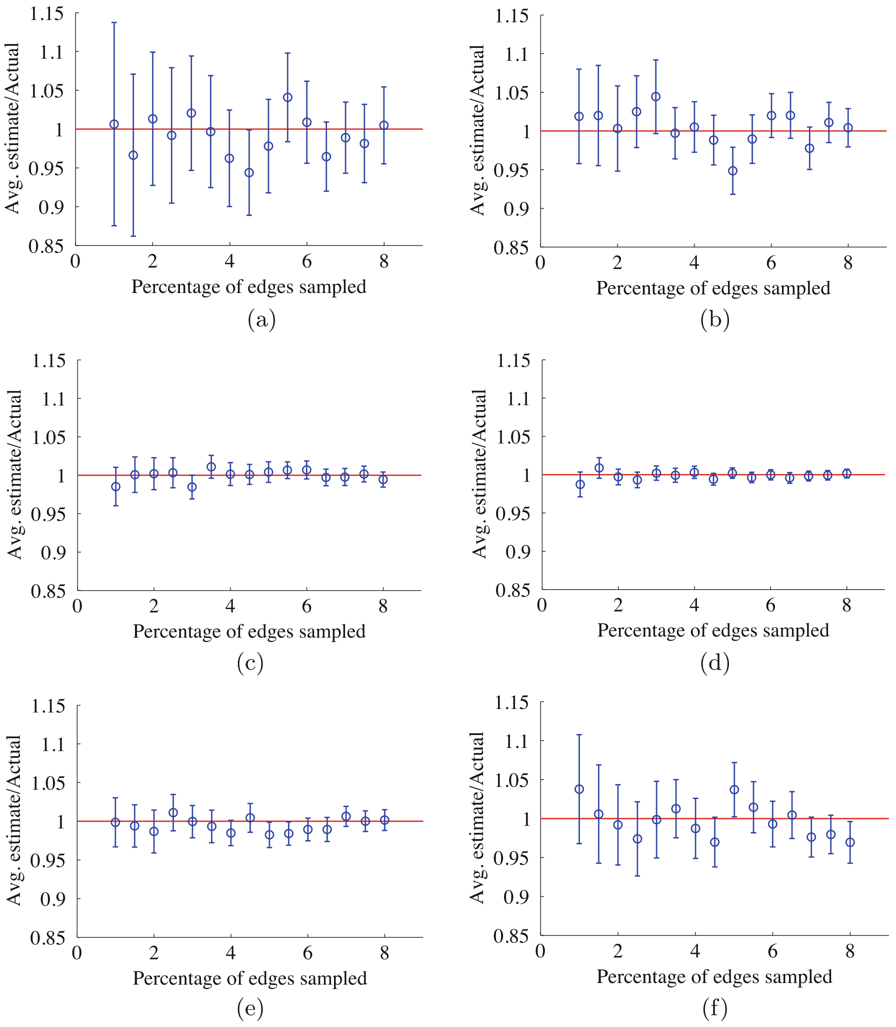


Fig. 6 Blue circles represent the ratio of the average estimated values of the total number of triangles to the actual value over 100 independent runs. Red line indicates 1. The blue error bars indicate 95% confidence intervals. **(a)** BA-1 ($\gamma = 1.5$, $N_T = 281,296$), **(b)** BA-2 ($\gamma = 1.5$, $N_T = 1,046,132$, **(c)** BA-3 ($\gamma = 1.5$, $N_T = 6,874,145$), **(d)** BA-4 ($\gamma = 1.0$, $N_T = 3,526,361$), **(e)** BA-5 ($\gamma = 1.5$, $N_T = 3,473,097$), **(f)** BA-6 ($\gamma = 2.0$, $N_T = 3,464,420$)

sampling fraction is used. By comparing Fig. 6a, b, and c where all of these figures are obtained by testing on BA graphs with the same power of the preferential attachment, we can see that the confidence intervals are larger in the BA graphs with a smaller number of triangles. In Fig. 6d, e, and f, the total number of triangles in each of the tested graphs is approximately equal but the values of γ and the global clustering coefficient are different. We can see that the estimate on graphs with a higher global clustering coefficient achieves a smaller confidence interval.

These results confirm the theoretical analysis of the relative error of the estimate presented in Sect. 3. Besides the sampling fraction, the relative error of the estimate is influenced by certain properties of the graph. Our algorithm achieves better accuracy on graphs with more triangles and a higher global clustering coefficient.

5 Conclusion

In this chapter, we propose a new framework for analyzing graphs in a dynamic system and present an edge sampling algorithm, called *Edge Sample and Discard* (ESD), which estimates the total number of triangles in a fully dynamic graph where both edge additions and deletions are possible. With a tiny modification of the weight parameter, ESD can also be applied to static graphs. Our algorithm achieves a significant improvement in accuracy by allowing the use of neighborhood information of the sampled edges through sending queries to the graph dataset. As illustrated in our performance analysis, ESD achieves much better accuracy, smaller variance, and faster speed than the previously known state-of-the-art algorithms under a slightly relaxed but practically relevant restrictions. In particular, it offers a methodology for the design of new algorithms in the future to keep track of the changes in the number of motifs of a certain type in a fully dynamic graph.

Acknowledgements This work was partially supported by the National Science Foundation Award #1250786.

References

1. Ahmed, N.K., Duffield, N., Neville, J., Kompella, R.: Graph sample and hold: a framework for big-graph analytics. In: ACM KDD, pp. 1446–1455. ACM, NY (2014)
2. Alon, N., Yuster, R., Zwick, U.: Finding and counting given length cycles. Algorithmica **17**(3), 209–223 (1997)
3. Avron, H.: Counting triangles in large graphs using randomized matrix trace estimation. In: Workshop on Large-Scale Data Mining: Theory and Applications, vol. 10, pp. 10–9 (2010)
4. Becchetti, L., Boldi, P., Castillo, C., Gionis, A.: Efficient algorithms for large-scale local triangle counting. ACM Trans. Knowl. Discov. Data **4**(3), 13 (2010)
5. Berry, J.W., Hendrickson, B., LaViolette, R.A., Phillips, C.A.: Tolerating the community detection resolution limit with edge weighting. Phys. Rev. E **83**(5), 056119 (2011)
6. Buriol, L.S., Frahling, G., Leonardi, S., Marchetti-Spaccamela, A., Sohler, C.: Counting triangles in data streams. In: ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, pp. 253–262. ACM, New York (2006)
7. Chakrabarti, D., Faloutsos, C.: Graph mining: laws, generators, and algorithms. ACM Comput. Surv. **38**(1), 2 (2006)
8. Chiba, N., Nishizeki, T.: Arboricity and subgraph listing algorithms. SIAM J. Comput. **14**(1), 210–223 (1985)

9. Coppersmith, D., Winograd, S.: Matrix multiplication via arithmetic progressions. In: Proceedings of the 19th Annual ACM Symposium on Theory of Computing, pp. 1–6. ACM, New York (1987)
10. Foucault Welles, B., Van Devender, A., Contractor, N.: Is a friend a friend?: investigating the structure of friendship networks in virtual worlds. In: CHI'10 Extended Abstracts on Human Factors in Computing Systems, pp. 4027–4032. ACM, New York (2010)
11. Gemulla, R., Lehner, W., Haas, P.J.: Maintaining bounded-size sample synopses of evolving datasets. VLDB J. **17**(2), 173–201 (2008)
12. Han, G., Sethu, H.: Edge sample and discard: a new algorithm for counting triangles in large dynamic graphs. In: IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining, pp. 44–49. ACM, New York (2017)
13. Hardiman, S.J., Katzir, L.: Estimating clustering coefficients and size of social networks via random walk. In: WWW, pp. 539–550. ACM, New York (2013)
14. Horvitz, D.G., Thompson, D.J.: A generalization of sampling without replacement from a finite universe. J. Am. Stat. Assoc. **47**(260), 663–685 (1952)
15. Jha, M., Seshadhri, C., Pinar, A.: A space-efficient streaming algorithm for estimating transitivity and triangle counts using the birthday paradox. ACM Trans. Knowl. Discov. Data **9**(3), 15 (2015)
16. Kolountzakis, M.N., Miller, G.L., Peng, R., Tsourakakis, C.E.: Efficient triangle counting in large graphs via degree-based vertex partitioning. Internet Math. **8**(1–2), 161–185 (2012)
17. Kutzkov, K., Pagh, R.: Triangle counting in dynamic graph streams. In: Algorithm Theory–SWAT 2014, pp. 306–318. Springer, Cham (2014)
18. Latapy, M.: Main-memory triangle computations for very large (sparse (power-law)) graphs. Theor. Comput. Sci. **407**(1), 458–473 (2008)
19. Leskovec, J., Krevl, A.: SNAP Datasets: Stanford large network dataset collection. http://snap.stanford.edu/data (Jun 2014)
20. Lim, Y., Kang, U.: Mascot: memory-efficient and accurate sampling for counting local triangles in graph streams. In: ACM KDD, pp. 685–694. ACM, New York (2015)
21. Newman, M.E.: The structure and function of complex networks. SIAM Rev. **45**(2), 167–256 (2003)
22. Pagh, R., Tsourakakis, C.E.: Colorful triangle counting and a MapReduce implementation. Inf. Process. Lett. **112**(7), 277–281 (2012)
23. Rossi, R.A., Ahmed, N.K.: The network data repository with interactive graph analytics and visualization (2013), http://networkrepository.com
24. Schank, T., Wagner, D.: Finding, counting and listing all triangles in large graphs, an experimental study. In: Experimental and Efficient Algorithms, pp. 606–609. Springer, Berlin (2005)
25. Shin, K.: Wrs: Waiting room sampling for accurate triangle counting in real graph streams. arXiv preprint arXiv:1709.03147 (2017)
26. Stefani, L.D., Epasto, A., Riondato, M., Upfal, E.: TRIÈST: Counting local and global triangles in fully-dynamic streams with fixed memory size. CoRR abs/1602.07424 (2016), http://arxiv.org/abs/1602.07424
27. Thorup, M., Zhang, Y.: Tabulation-based 5-independent hashing with applications to linear probing and second moment estimation. SIAM J. Comput. **41**(2), 293–331 (2012)
28. Tiropanis, T., Hall, W., Crowcroft, J., Contractor, N., Tassiulas, L.: Network science, web science, and Internet science. Commun. ACM **58**(8), 76–82 (2015)
29. Tsourakakis, C.E.: Fast counting of triangles in large real networks without counting: algorithms and laws. In: 2008 8th IEEE International Conference on Data Mining, pp. 608–617. IEEE, Pisa (2008)
30. Tsourakakis, C.E., Kang, U., Miller, G.L., Faloutsos, C.: Doulion: Counting triangles in massive graphs with a coin. In: ACM KDD, pp. 837–846, ACM, New York (2009)
31. Türkoğlu, D., Turk, A.: Edge-based wedge sampling to estimate triangle counts in very large graphs. arXiv preprint arXiv:1710.09961 (2017)

32. Wasserman, S., Faust, K.: Social Network Analysis: Methods and Applications, vol. 8. Cambridge University Press, Cambridge (1994)
33. Welser, H.T., Gleave, E., Fisher, D., Smith, M.: Visualizing the signatures of social roles in online discussion groups. J. Soc. Struct. **8**(2), 1–32 (2007)
34. Yahoo! webscope dataset. http://research.yahoo.com/Academic_Relations

Generation and Corruption of Semi-Structured and Structured Data

Samir Al-janabi and Ryszard Janicki

Abstract It is crucial for data to be a reliable source of information so that decisions made based on the analysis of this data could provide a competitive edge and reduce the negative impacts that pose significant cost to organizations on an annual basis. This data could have more than one form, including that both of semi-structured and structured data. There are many factors that could corrupt and cause degradation in the quality of data including duplicate records, inaccurate values, inconsistent values, outdated data, or incomplete information. To maintain the quality of data, the algorithms of different data quality management approaches need to be compared, and to accomplish this, common datasets need to be presented. These datasets could be real or synthetic. In the latter type, the datasets need to satisfy intrinsic characteristics of data. However, such datasets are not common for reasons such as privacy constraints in the case of real datasets, or the synthetic data that is generated or corrupted by the existing systems may not satisfy the quality aspects. To address these issues, we present a system that allows for generation of semi-structured and structured data. The generated semi-structured data is XML documents and the generated structured datasets satisfy a set of integrity constraints. Also our system generates other data values such as personal data and sensors data. Additionally, it allows for the corruption of the generated semi-structured and structured data.

1 Introduction

Data is used in various organizational activities such as identifying patterns in consumer habits, or determining what new product or service to create next. There is tremendous value in data and consequently, having the knowledge to improve data quality most efficiently is essential. Users and companies at large need to be able to depend on the information that the data produces so that it is a reliable source of

S. Al-janabi (✉) · R. Janicki
McMaster University, Hamilton, ON, Canada
e-mail: aljanasa@mcmaster.ca; janicki@mcmaster.ca

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_7

information for analysis. Organizations use data to support different activities, and researchers in data quality management use data to validate hypotheses and evaluate the quality and scalability of their proposed algorithms. The area of data quality is thus of concern to both practitioners and researchers [1]. The development of effective systems that detect and correct data issues is becoming an important factor for managing the quality of data. Problems related to data quality cost businesses in the USA more than $600 billion a year [2]. There is an increased demand in the industries for developing data quality management systems [3]. There are five central data quality aspects. It is necessary to deal with each one of these elements, in order to improve the quality of data [4].

There is more than one type of data. Data types include structured, semi-structured, and unstructured data. Structured data is represented by a strict format, that is, there is a formal structure of data models associated with that data. An example of this type is the data stored in relational databases. The DBMS checks to ensure that the data stored conforms to the specified formal structure and constraints [5]. Constraints can be expressed in this type of data as functional dependencies (FDs), conditional functional dependencies (CFDs), and inclusion dependencies (INDs). Semi-structured data allows for some degree of flexibility in the structure of data [6]. The data may have a certain structure but it is not necessary for all data to have an identical structure. Some attributes may exist only in some entities. This type of data is commonly referred to as self-describing documents. The data typically contains the information that is normally associated with a schema [7]. Examples of this type of data are XML (extensible markup language) and JSON (JavaScript Object Notation). There are different uses for XML documents, such as exchanging data over the Web in text files. Two main concepts are used in the structure of XML documents: elements and attributes. Simple elements contain data values, while hierarchically, complex elements are constructed from other elements [8]. With unstructured data, there is no defined structure and data is expressed in natural language[6]. An example of unstructured data is when there is a text document which contains information embedded within it such as a web page in HTML. Rather than describe the meaning of various data elements in the document, HTML documents describe the format of the document through tags that describe how it is displayed. In this work, we generate and corrupt semi-structured and structured data.

To evaluate their algorithms and techniques, data scientists, researchers, analysts, and engineers working in data analytics, databases, information retrieval, or machine learning require data with certain properties. A plentiful number of techniques have been proposed to fix data quality problems such as [9–11], but to do representative comparisons, a common gold standard is required. However, knowledge of duplicate records, that is, the gold standard, can be difficult to obtain. Real datasets are usually unavailable due to confidentiality and privacy constraints that make it difficult for data to be published [12, 13]. Providing a common ground for existing algorithms to detect and correct errors in duplicate records is essential. However, comparing between these algorithms, with the exception of a few datasets, is not allowed. One of the main reasons for this is the lack of common datasets [14].

To tackle these challenges, we present datumPIPE 2.0, a **d**ata gener**at**or and corr**u**pter of **m**ulti**P**le data qual**I**ty as**PE**cts—version **2.0**, an extended version of the system in [15]. The key features of datumPIPE 2.0 are: 1. it generates semi-structured data represented by XML documents; 2. it corrupts the XML documents by introducing errors in terms of data accuracy and information completeness of data quality aspects; 3. it generates structured data that satisfy different types of integrity constraints including FDs, constant CFDs, variable CFDs, and INDs; 4. it corrupts the record of the structured data through corruption of the central five aspects of data quality including data deduplication, data consistency, information completeness, data accuracy, and data currency; 5. it generates other types of attributes values in the data such as joint values, sensors data, and personal data ranging from names and dates, to bank account numbers and SSNs, as well as numerical values; and 6. it marks the original generated records of the data.

2 Overview of the System

The system architecture of datumPIPE 2.0 is illustrated in Fig. 1. It consists of two main parts: one part generates data and other part corrupts data, where the data models are based on the XML data model (also called hierarchical model or tree model) and the relational data model, for the semi-structured and structured data, respectively.

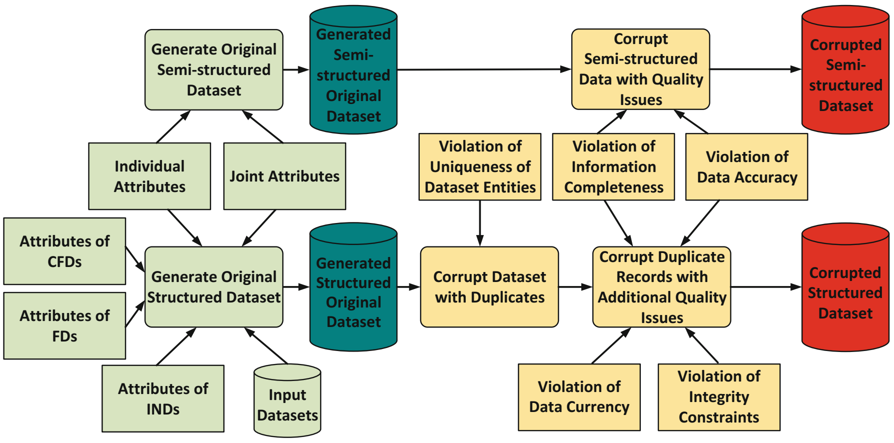

Generate Original Semi-structured Dataset
Generated Semi-structured Original Dataset
Corrupt Semi-structured Data with Quality Issues
Corrupted Semi-structured Dataset
Individual Attributes
Joint Attributes
Violation of Uniqueness of Dataset Entities
Violation of Information Completeness
Violation of Data Accuracy
Attributes of CFDs
Attributes of FDs
Generate Original Structured Dataset
Generated Structured Original Dataset
Corrupt Dataset with Duplicates
Corrupt Duplicate Records with Additional Quality Issues
Corrupted Structured Dataset
Attributes of INDs
Input Datasets
Violation of Data Currency
Violation of Integrity Constraints

Fig. 1 datumPIPE 2.0 System Architecture

2.1 Generation of Semi-Structured Data

The attributes in an XML document can be used to provide additional information that describes elements. An XML document can be a schemaless XML document when it does not conform to a predefined XML schema in which it is considered semi-structured data. If it conforms to a predefined XML schema such as DTD document, it is considered to be structured data. The attribute term in an XML document is used differently than in relational databases. In XML, it is used as a description language [8]. datumPIPE 2.0 generates semi-structured data type. The elements in an XML document are identified by their start tag and end tag. The tag names are enclosed between angled brackets < ... >, and end tags are identified by a slash </ ... >. The component *Generate Original Semi-structured Dataset* is used to generate the *Generated Semi-structured Original Dataset*. datumPIPE 2.0 allows for the generation of the following attributes:

Individual Attributes datumPIPE 2.0 generates the attribute values of XML data using the individual attributes component in Fig. 1. Figure 2 illustrates some generated individual attributes. For example, the complex elements are the ones with the tag names <Employees> and<Employee>, while the simple elements are the ones with the tag names such as <RecordID>, <BankAccount>, <SSN>, <Phone>, <Bin8>, <Bin16>, <FirstName>, <LastName>, <Email>, <RandomInteger>, <Date>,<DateRange>, and <SequentialInteger>. datumPIPE 2.0 can also generate other attributes that are not shown in Fig. 2.

Joint Attributes datumPIPE 2.0 generates the joint attribute values using the joint attributes component in Fig. 1. Figure 2 illustrates a generated joint attribute <Email>. This attribute value is the concatenation of <FirstName> and <Company> attribute values. For example, in line 30 of Fig. 2, the value of the email is "adrienne@proinindustries.com," which is based on the first name "Adrienne" and the company name "Proin Industries.'

2.2 Corruption of Semi-Structured Data

Original semi-structured generated data are corrupted using the *Corrupt Semi-structured Data with Quality Issues* component. The user could specify different parameter settings to corrupted XML data such as the rate of corruption in the XML attribute values. The attribute value <RecordID> does not get corrupted. datumPIPE 2.0 allows for the corruption of two types of data quality aspects:

Inaccurate Data Errors are introduced in the attribute values in terms of typographical errors including deletion of a character, insertion of a new character, substitution of a character with a new character, or transposition of two adjacent characters. In addition, swapping of values for each attribute is introduced. Figure 3 illustrates some corrupted attribute values. For example, in the <FirstName> tag in

```
 1: <?xml version="1.0 standalone="yes?>
 2: <Employees>
 3: <Employee>
 4:      <RecordID>53</RecordID>
 5:      <BankAccount>3-562-3917129</BankAccount>
 6:      <SSN>591-635-525</SSN>
 7:      <Phone>1010011</Phone>
 8:      <Bin8>011111101000010</Bin8>
 9:      <Bin16>11000100010111100111001111100100</Bin32>
10:      <FirstName>Adam</FirstName>
11:      <LastName>Marc</LastName>
12:      <Company>Et Ipsum Ltd</Company>
13:      <Email>adam@etipsumltd.net</Email>
14:      <RandomInteger>212 </RandomInteger>
15:      <Date>2016/12/10< /Date>
16:      <DateRange>1995/7/14</DateRange>
17:      <SequentialInteger>153</SequentialInteger>
18:      <RandomDecFraction>0.6267489831533459</RandomDecFraction>
19: </Employee>
20: <Employee>
21:     <RecordID>96</RecordID>
22:      <BankAccount>2-405-4686184</BankAccount>
23:      <SSN>545-682-889</SSN>
24:      <Phone>+1(950)-722-1807</Phone>
25:      <Bin8>0010111</Bin8>
26:      <Bin16>001000100111100</Bin32>
27:      <FirstName>Adrienne</FirstName>
28:      <LastName>Joseph</LastName>
29:      <Company>Proin Industries</Company>
30:      <Email>adrienne@proinindustries.com</Email>
31:      <RandomInteger>204</RandomInteger>
32:      <Date>1989/3/23</Date>
33:      <DateRange>1995/4/25</DateRange>
34:      <SequentialInteger>196</SequentialInteger>
35:      <RandomDecFraction>0.8097480696988298</RandomDecFraction>
36: </Employee>
37: </Employees>
```

Fig. 2 An instance of a generated XML document

line 11, the value is "Mabrc" instead of "Marc" which represents a transcription error in terms of insertion of a new character. In the <Company> tag in line 29, the value is "Industries Proin" instead of "Proin Industries," thus indicating a swap of values of the tag.

Incomplete Information Errors are introduced in the attribute values in terms of incomplete information by making them missing, i.e., NULL. Figure 3 illustrates some corrupted attribute values. For example, in the <LastName> tag in line 28, the value is missing.

```
<?xml version=“1.0 standalone=“yes?>
<Employees>
<Employee>
    <RecordID>53</RecordID>
    <BankAccount>3-562-3917129</BankAccount>
    <SSN>591-635-525</SSN>
    <Phone>+1(825)-614-3515</Phone>
    <Bin8>1010011</Bin8>
    <Bin16>011111101000010</Bin32>
    <FirstName>Avdam</FirstName>
    <LastName>Mabrc</LastName>
    <Company>Et Ipsm Ltd</Company>
    <Email>adam@etipbsumltd.net</Email>
    <RandomInteger>212</RandomInteger>
    <Date>2016/12/10< /Date>
    <DateRange>1995/7/14 </DateRange>
    <SequentialInteger>153</SequentialInteger>
    <RandomDecFraction>0.6267489831533459</RandomDecFraction>
</Employee>
<Employee>
   <RecordID>96</RecordID>
    <BankAccount>2-405-4686184</BankAccount>
    <SSN>545-682-889</SSN>
    <Phone>+1(950)-722-1807</Phone>
    <Bin8>0010111</Bin8>
    <Bin16>001000100111100</Bin32>
    <FirstName>Adrienne</FirstName>
    <LastName> </LastName>
    <Company>Industries Proin</Company>
    <Email>adrienne@proinindustries.com</Email>
    <RandomInteger>204</RandomInteger>
    <Date>1989/3/23< /Date>
    <DateRange>1995/4/25</DateRange>
    <SequentialInteger>196</SequentialInteger>
    <RandomDecFraction>0.8097480696988298</RandomDecFraction>
</Employee>
</Employees>
```

Fig. 3 An instance of a corrupted XML document

2.3 Generation of Structured Data

The component *Generate Original Structured Dataset* is used to generate the *Generated Structured Original Dataset* in which different types of attributes can be generated. The user specifies different parameter settings to generate records such as the number of required generated records, attributes types, and the dependency between attributes. The generated records are marked through the attribute ID. datumPIPE 2.0 allows for the generation of the following types of attributes:

Table 1 An instance of a generated original dataset that satisfies an FD

	ID	FName	City	Temp
t_1	458	Penny	Sun City	35
t_2	4670	Ray	Sun City	35
t_3	738	Eaton	Summerfield	42
t_4	2452	Vickie	Summerfield	42
t_5	3167	Myrtle	Summerfield	42
t_6	2494	Michael	Pine Plains	29

Table 2 An instance of a generated original dataset that satisfies a variable CFD

	ID	Rank	Salary
t_1	4036	e	1928
t_2	4037	d	2113
t_3	4038	e	1928
t_4	4039	c	2368
t_5	5926	d	2113
t_6	5947	c	2368

Table 3 An instance of a generated original dataset that satisfies a constant CFD

	ID	Model	Car
t_1	7284	Ranger	Ford
t_2	7666	Ranger	Ford
t_3	3656	Venza	Toyota
t_4	4727	Venza	Toyota

FD Attributes Table 1 shows data generated by datumPIPE 2.0. There is a functional dependency ϕ_1: [City] $\rightarrow$ [Temp] in which the attribute value of $[X]$ determines the attribute value of $[Y]$.

CFD Attributes Table 2 shows data generated by datumPIPE 2.0. There is a variable CFD, ϕ_2: ([Rank] $\rightarrow$ [Salary], T_1), where T_1 states that if the value of the rank of any two ranks is "e," then the value of the salary should be the same, and so on for other rank values.

Table 3 shows data generated by datumPIPE 2.0. There is a constant CFD, ϕ_3: ([model] $\rightarrow$ [car], T_2), where T_2 states that if the value of the model is "Ranger," then the value of the car should be "Ford," and so on for other pairs of model and car values.

IND Attributes Table 4 shows data generated by datumPIPE 2.0. The attribute values are taken from another dataset in the *Input Datasets* component.

Table 4 An instance of a generated original dataset that satisfies an IND

	ID	dept
t_1	3993	dept3
t_2	3994	dept5
t_3	3995	dept5
t_4	3996	dept4

Table 5 An instance of data with inconsistent values

	ID	City	Temp
t_1	2452	Summerfield	42
t_2	2452	Summerfield	39
t_3	2452	Summerfield	42
t_4	2463	Forest Hill	38
t_5	2463	Forest Hill	27
t_6	2463	Forest Hill	38

2.4 Corruption of Structured Data

Original structured generated records are corrupted using *Corrupt Dataset with Duplicates* and *Corrupt Duplicate Records with Additional Quality Issues* components. Different parameter settings to corrupted records can be specified by the user such as the rate of corruption in the duplicated records, what dependencies to corrupt, and the percentage of duplicated records. The attribute value ID does not get corrupted so the corrupted records that belong to the same entity are known. datumPIPE 2.0 allows for the corruption of five types of data quality aspects:

Duplicate Records datumPIPE 2.0 introduces errors in the generated data through duplicating the original generated records. The percentage of the records to duplicate in the original records and the number of duplicates per records can be specified by the user.

Inaccurate Data Errors are introduced in the attribute values in similar techniques to that in Sect. 2.2.

Inconsistent Data datumPIPE 2.0 introduces errors in the generated data in terms of inconsistency through corrupting the left side attribute in the IND and corrupting the right side of the FD and the CFD. Corruption could include typographical errors, as well as swapped or missing values. Table 5 shows a corrupted attribute value t_2[Temp] that makes ϕ_1: [City] $\rightarrow$ [Temp] unsatisfied.

Incomplete Information datumPIPE 2.0 introduces errors in the generated attribute values in terms of information incompleteness by making them missing, i.e., NULL. We assume a model with NULL marks with a closed world assumption (CWA) where some attribute values of a tuple could be missing (NULL). In Table 6, the attribute value t_2[Email] is missing.

Table 6 An instance of data with incomplete information

	ID	Email
t_1	5451	jason@vulputatellc.org
t_2	5451	NULL
t_3	5451	jason@vulputatellc.org
t_4	5458	bernice@sitllc.net

Table 7 An instance of data with outdated values

	ID	Date
t_1	4791	12/1/1995
t_2	4791	12/1/1995
t_3	4791	7/26/1970
t_4	4792	7/3/1950

Data Currency datumPIPE 2.0 introduces errors in the generated attribute values in terms of corruption of the timestamps in the original records through introducing corrupted values. In Table 7, the attribute value t_3[Date] is corrupted.

Other Types of Attributes Other types of attribute values can be generated by datumPIPE 2.0 such as joint attributes in which two different attribute values are joint into one value. For example, the attribute Email with a value of "anne@etpc.net" is a join of the generated attribute values in first name (FName) and the company (Company). In addition, datumPIPE 2.0 can generate other types of attribute values such as sensors binary data, SSN, bank account numbers, dates, random integers, and sequential integers.

The *Input Datasets* component of datumPIPE 2.0 also represents all the datasets that are used to generate some of the values such as names of persons and cities. In addition, some attribute values including SSN, bank account numbers, and emails are generated uniquely. We also assume in this version of datumPIPE 2.0 that the sets of X and Y of the integrity constraints attributes are singleton and disjoint sets.

3 Related Work

Different frameworks and tools have been developed to generate semi-structured and structured data and/or to corrupt this data. However, most of them neither generate and corrupt XML documents nor corrupt structured data. Also, they do not provide generation of records that satisfy sets of FDs, CFDs, and INDs and do not corrupt multiple data quality aspects. In TPC-H Benchmark, almost all the generated columns, with few exceptions, are uncorrelated, and there is no corruption functionality (http://www.tpc.org/tpch/default.asp). Other online data generation or commercial tools (e.g., http://www.generatedata.com or http://www.sqledit.com) lack the corruption functionality and the generation of structured data with integrity constraints including FDs, CFDs, and INDs. In the work of [16], a data generator for XML collections is developed. In the work of [17], a synthetic XML data generator is developed that generates data with regards to a given set of XML queries expressed in XPath. Tran [18] introduced a tool to generate and corrupt data in which the generated attribute values can be compound where the values depend on each other or are independent of each other. The tool corrupts datasets with missing values and typographical errors and some other types of errors. The tool of [19] generates data with attribute dependencies. In the work of [20], data mining techniques are used to help generating synthetic data. Lin [21] introduced a tool to generate data in order to test data mining tools. A tool developed by [22] generates synthetic trajectory datasets in order to simulate mobility behaviors. An interactive tool developed by [23] generates data for the merge/purge problem. It allows the user to set the percentage of duplicate records in the data, the size of data, and the error rate. The types of errors include swapping of values and typographical errors. The generated fields include data such as cities, states, zip codes, and names. Christen [24] developed a system named febrl that can generate data. This system may include look-up files with name variations and frequency distribution tables. In the context of ETL processes, [25] presented a work for data generation. It aims to evaluate the quality characteristics in addition to the correctness of ETL process models.

4 Conclusions and Future Work

datumPIPE 2.0 is a tool that can be used to generate semi-structured datasets in terms of XML documents and to generate structured datasets that satisfy multiple data quality aspects. In addition, the generated semi-structured and structured datasets can be corrupted. It is an interactive tool in which the user can specify different parameter settings for the generation and corruption. Different applications can benefit from this tool, such as algorithm evaluation when there is a need for a dataset that satisfies specific data models and types. In future work, we plan to have an online graphical user interface, to generate big data, and to generate other data models, such as in the JSON data format.

References

1. Watts, S., Shankaranarayanan, G., Even, A.: Data quality assessment in context: a cognitive perspective. Decis. Support. Syst. **48**(1), 202–211 (2009)
2. Eckerson, W.: Data Quality and the Bottom Line: Achieving Business Success Through a Commitment to High Quality Data, pp. 1–36. The Data Warehousing Institute, Renton (2002)
3. Judah, S., Friedman, T.: Magic Quadrant for Data Quality Tools. Technical Report. Gartner, Stamford (2014)
4. Fan, W., Geerts, F.: Foundations of Data Quality Management. Morgan & Claypool Publishers, San Rafael (2012)

5. Silberschatz, A., Korth, H.F., Sudarshan, S.: Database System Concepts. McGraw-Hill, New York (2006)
6. Batini, C., Scannapieca, M.: Data Quality Concepts, Methodologies and Techniques. Springer, New York (2006)
7. Buneman, P.: Semistructured Data. In: PODS '97 Proceedings of the Sixteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, pp. 117–121. ACM, New York (1997)
8. Elmasri, R., Navathe, S.B.: Fundamentals of Database Systems. Pearson, Boston (2015)
9. Al-janabi, S., Janicki, R.: A density-based data cleaning approach for deduplication with data consistency and accuracy. In: SAI Computing Conference (SAI), pp. 492–501. IEEE, Piscataway (2016)
10. Cao, Y., Fan, W., Yu, W.: Determining the Relative Accuracy of Attributes. In: Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data, pp. 565–576. ACM, New York (2013)
11. Fan, W., Geerts, F., Tang, N., Yu, W.: Conflict resolution with data currency and consistency. J. Data Inf. Qual. **5**(1–2), 6 (2014)
12. Christen, P.: Data Matching Concepts and Techniques for Record Linkage, Entity Resolution, and Duplicate Detection. Springer, Canberra (2012)
13. Naumann, F., Herschel, M.: An Introduction to Duplicate Detection. Morgan & Claypool Publishers, San Rafael (2010)
14. Weis, M., Naumann, F., Brosy, F.: A duplicate detection benchmark for XML (and relational) data. In: SIGMOD Workshop on Information Quality for Information Systems (IQIS) (2006)
15. Al-janabi, S., Hamid, A., Janicki, R.: datumPIPE: data generator and corrupter for multiple data quality aspects. In: Proceedings of the 2017 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2017, pp. 589–592. ACM, New York (2017)
16. Pérez, M., Sanz, I., Berlanga, R.: XTaGe: A flexible XML collection generator. In: Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data, pp. 1139–1142. ACM, New York (2010)
17. Rychnovský, D., Holubová, I.: Generating XML data for XPath queries. In: Proceedings of the 30th Annual ACM Symposium on Applied Computing, pp. 724–731. ACM, New York (2015)
18. Tran, K.-N., Vatsalan, D., Christen, P.: GeCo: an online personal data generator and corruptor. In: Proceedings of the 22nd ACM International Conference on Information & Knowledge Management, pp. 2473–2476. ACM, New York (2013)
19. Houkjær, K. , Torp, K., Wind, R.: Simple and realistic data generation. In: Proceedings of the 32nd International Conference on Very Large Data Bases, pp. 1243–1246. VLDB Endowment (2006)
20. Eno, J., Thompson, C.: Generating synthetic data to match data mining patterns. IEEE Internet Comput. **12**(3), 78–82 (2008)
21. Lin, P., Samadi, B., Cipolone, A., Jeske, D., Cox, S., Rendón, C., Holt, D., Xiao. R.: Development of a synthetic data set generator for building and testing information discovery systems. In: Third International Conference on Information Technology: New Generations, 2006. ITNG 2006, pp. 707–712. IEEE, Piscataway (2006)
22. Pelekis, N., Sideridis, S., Tampakis, P., Theodoridis, Y.: Hermoupolis: a semantic trajectory generator in the data science era. SIGSPATIAL Spec. **7**(1), 19–26 (2015)
23. Hernández, M., Stolfoz, S.: The merge/purge problem for large databases. In: Proceedings of the 1998 ACM-SIGMOD Conference (1995)
24. Christen, P.: Development and user experiences of an open source data cleaning, deduplication and record linkage system. SIGKDD Explor. **11**(1), 39–48 (2009)
25. Nakuçi, E., Theodorou, V., Jovanovic, P., Abelló, A.: Bijoux: data generator for evaluating ETL process quality. In: Proceedings of the 17th International Workshop on Data Warehousing and OLAP, pp. 23–32. ACM, New York (2014)

A Data Science Approach to Predict the Impact of Collateralization on Systemic Risk

Sharyn O'Halloran, Nikolai Nowaczyk, Donal Gallagher, and Vivek Subramaniam

Abstract In this chapter, we simulate and analyze the impact of financial regulations concerning the collateralization of derivative trades on systemic risk—a topic that has been vigorously discussed since the financial crisis in 2007/08. Experts often disagree on the efficacy of these regulations. Compounding this problem, banks regard their trade data required for a full analysis as proprietary. We adapt a simulation technology combining advances in graph theory to randomly generate entire financial systems sampled from realistic distributions with a novel open-source risk engine to compute risks in financial systems under different regulations. This allows us to consistently evaluate, predict, and optimize the impact of financial regulations on all levels—from a single trade to systemic risk—before it is implemented. The resulting data set is accessible to contemporary data science techniques like data mining, anomaly detection, and visualization. We find that collateralization reduces the costs of resolving a financial system in crisis, yet it does not change the distribution of those costs and can have adverse effects on individual participants in extreme situations.

Keywords Big data · Graph theoretic models · Data science · Machine learning · Python · C++ · Random graph generation · Stochastic Linear Gauss-Markov model · Monte Carlo simulation · Financial risk analytics · Systemic risk · Collateralizations · Variation margin · Initial margin · Open-source risk engine · Financial regulation

S. O'Halloran · V. Subramaniam
Columbia University, New York, NY, USA
e-mail: so33@columbia.edu; vs2575@columbia.edu

N. Nowaczyk (✉) · D. Gallagher
Quaternion Risk Management, London, UK
e-mail: nikolai.nowaczyk@quaternion.com; donal.gallagher@quaternion.com

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_8

1 Introduction

Counterparty credit risk (CCR) is the risk of suffering a loss when a contracting party defaults before satisfying its obligations. While banks and financial institutions have been well aware of the CCR in classical business lines like loans for centuries, the credit risk component in derivative trades has gained recent attention. Since the 2008 crisis, markets can no longer assume that the credit risk in a derivative with a bank is zero—even if it is AAA rated. As the volume of the over-the-counter (OTC) derivative business alone exceeds \$20,701 billions of the US dollars,[1] it has been suggested that the inherent credit risk now poses a significant threat to the system. Indeed, a number of financial regulations have been enacted aimed at reducing this credit risk. A key feature of these regulations is that they provide strong incentives (and increasingly the obligation) to collateralize derivative trades, i.e., post and receive capital.

It is obvious that a counterparty[2] that receives collateral for its derivative trades has a lower CCR exposure than a counterparty that does not. However, it is less obvious to show that the introduction of collateralization reduces the systemic risk in a financial system as a whole. Since the turmoil following the collapse of Lehman Brothers in the 07/08 crisis, there has been a vibrant discussion on how to improve financial regulation to avoid another crisis in the future. This crisis has not only questioned the soundness of the current regulation but also the process on how to decide what regulations to implement. Although a decade has passed by now, there is still little consensus on how to evaluate, predict, and optimize financial regulation, that is:

1. Evaluate: Has the regulation implemented since the financial crises reduced systemic risk or not?
2. Predict: How can we predict the impact of a financial regulation before it is implemented on the real financial system?
3. Optimize: How can we find the best possible financial regulation?

Data science provides new solutions to measuring and predicting systemic risk in financial systems. It is the aim of this article to use these new technologies to develop a framework in which these questions can be answered quantitatively and systematically and to apply this technology to prominent regulations regarding the collateralization of derivative trades.

The rest of this chapter is organized as follows: In Sect. 2, we provide a framework to coherently extend enterprise-level risk metrics to a systemic level using a graph model, in which the nodes represent individual banks and the links represent trade relations. We present a purely graph theoretic technique on how

[1]The corresponding outstanding notional amount of all contracts in 2016 was \$544,052 billions of the US dollars. See Bank of International Settlements, Global OTC Derivatives Market Semi-Annual Statistics, March 6, 2017. http://stats.bis.org/statx/srs/table/d5.1.

[2]In this chapter, we assume all our counterparties to be banks.

to aggregate the risks arising in those trades to a counterparty and to a systemic level employing a weighted multivariate version of the in/out degree. In Sect. 3, we present in detail our data science approach to predict the impact of a regulation on systemic risk using recent advances in random graph generation and a novel open-source risk engine (ORE). We present our results in Sect. 4 and conclude with further research plans in Sect. 5.

The analysis validates that collateralization does indeed reduce systemic risk; it does not, however, change the distribution of risk in a system. The analysis also illustrates that validation depends on the metric chosen to quantify systemic risk. We find that while on a macro-level, the impact of collateralization is unambiguously risk reducing, and it can increase the risk stemming from individual netting sets in extreme cases.

1.1 State of the Art

One of the main methods currently used to evaluate financial regulation is expert judgment (of course based on quantitative studies providing a snapshot of data of the current financial system). The main disadvantage of expert judgment is that different experts have different opinions. If experts do not even agree in their judgment on whether or not the changes in regulations so far are improvements, there is little hope to answer the question of how to find an optimal financial regulation with that method.

Data science and ultimately machine learning offer a viable alternative by tackling some of the root causes why experts fail to agree:

1. Micro- vs. Macro-regulation: In financial economics, the impact of a financial regulation is currently studied either on the *micro-prudential* level, which considers only a single individual bank with its trades in all its detail, but ignores systemic effects, or on the *macro-prudential* level, which considers only systemic effects, but ignores most of the implications on individual banks, see Fig. 1. As financial regulations heavily affect the financial system on both levels—sometimes in opposite ways—this gap makes it very difficult to arrive at consistent answers.
2. Inconsistent metrics and unclear definitions: This gap is particularly visible when reviewing risk metrics. On the micro-prudential level, there are very well-defined types of risk, for instance *market risk* (suffering a loss due to adverse market movements), *counterparty credit risk* (suffering a loss due to a default of a business partner), *liquidity risk* (running out of money), or *model risk* (models used to quantify any of the previous risks are wrong). Furthermore, there are standardized metrics to measure them like *Value-at-Risk*, *Effectivized Expected Positive Exposure (EEPE)*, *Liquidity Coverage Ratio*, or a *Basel Traffic Light Test*. Not only are these metrics standardized, their use is enforced globally by regulators.

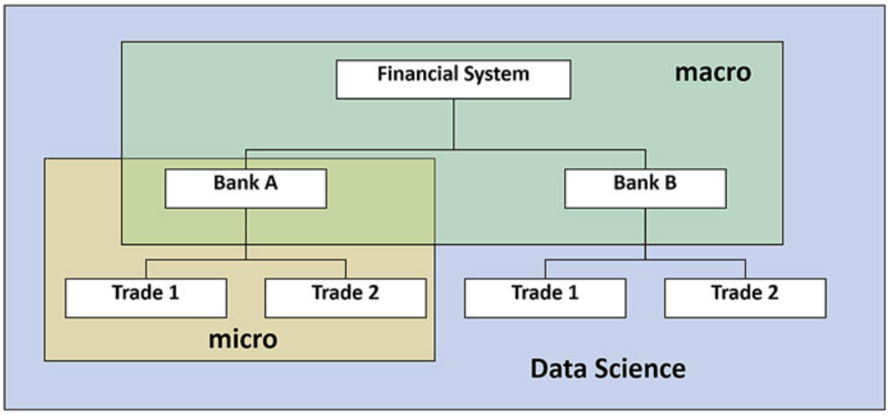


Fig. 1 Using data science to close the gap between micro- and macro-prudential regulation

On the macro-prudential level, there is not even a clear definition on what systemic risk means or how it should be measured. In [7], the US *Office for Financial Research* discusses 31 metrics of systemic risk. Most of these metrics focus on the analysis of market data like housing prices or government bonds and their correlations. For instance, [6] use principal components analysis (PCA) and Granger causality to study the correlations between the returns of banks, asset managers, and insurances. Unfortunately, most of those macro-prudential metrics are unsuited to guide decision-making bodies or regulatory interventions—precisely because their micro-prudential nature remains unclear (with CoVaR, which relies on a quantile of correlated asset losses, being a notable exception, see [1]).

This incoherence in metrics yields to incompatible views when evaluating the impact of financial regulations. One of the key strengths of a data science approach is that it enforces a consistent quantification of features and transparent evaluation metrics.

3. Lack of Data: While macro-economic data of the financial markets is easily available, even macro-economic data on the financial system is often not publicly available, Cont et al. were able to obtain macro-level data of aggregate interbank exposures of the Brazilian banking system during 2007/08 from the Brazilian Central Bank and adopt a graph model to describe the interconnectedness of the system and to estimate the impact that an increase in capital requirements has on those interbank exposures, see [10].

   While those analyses are very interesting on a macro-prudential level, they cannot predict the impact of financial regulations on a micro-prudential level, because their impacts depend on a banks' individual trades. However, all banks regard their trade data as proprietary.

1.2 Method and Approach

The lack of data is a nontrivial practical problem for the calibration and backtesting of any model of systemic risk. Notice that even if one would obtain the trade data of the entire financial system as of today and test the impact of a regulation with the aim of guiding a decision-making process, this would still not guarantee that its implementation has the desired effects. The regulations regarding Initial Margin are a consequence of the financial crisis in 2007/08, but the last phase-in date for Initial Margin is currently set at 2020. A lot of the trades that existed in 2007 will have matured in 2020. Therefore, it is imperative to make sure that the regulation is future proof and has the desired effect not only on the current financial system but ideally also on all possible future versions of it.

Our approach is to randomly generate financial systems sampled from realistic distributions and simulate their future evolution under different regulatory regimes. The key tool is a systemic risk engine, which combines a novel open-source risk engine with graph theoretic models. This allows us to consistently calculate the risks of all individual banks and the system as a whole, hence bridging the gap between micro- and macro-prudential economics.[3]

The systemic risk engine uses the following key components and steps, see also Fig. 2:

1. Graph Model—Trade Relations: Any financial system at a fixed point in time can be modeled using an undirected graph, where the nodes represent the banks and the links represent the trade relations between them, see Definition 1 for details and Fig. 3 for a simplified example.[4]
2. Graph Model—Risk Metrics: The exposures stemming from the trade relations in a financial system can be represented as a directed graph where the nodes again represent the same banks and the arrows represent the risk induced from the tail to the head, see Definition 2 for the details and Fig. 4 for an example. The standardized micro-prudential metrics give rise to weight functions on the arrows, which quantify the risks stemming from the trade relations, for instance EEPE for counterparty credit risk. These can then be aggregated consistently to systemic versions using weighted in and out degrees, see Definition 3 and Definition 4 for details.
3. Financial System Generation: We randomly generate trade relation graphs $G_1, \dots, G_s$, of financial systems sampling from the distributions found in [10], see Fig. 5 for an example. This is achieved using the configuration erase model from the `Python` library `networkx` based on [5, 8, 16], see component I in Fig. 2.

[3]In an earlier paper, see [18], we examined a case study that analyzed a single financial system under varying regulatory regimes, see also.

[4]In reality, these graphs are significantly more complex. According to an analysis carried out in [10] based on central bank data, the Brazilian financial system for example has about 2400 banks heavily interconnected via 20,000 links.

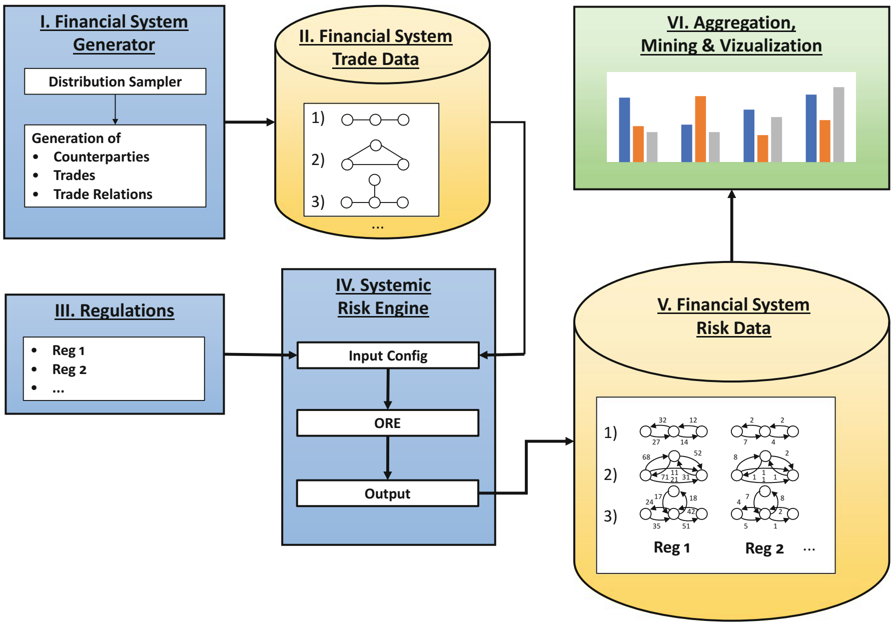


Fig. 2 IT architecture of data science solution

Fig. 3 Trade relations: the nodes represent the banks, the links represent the trade relations, and the labels on the links represent the trade or portfolio IDs

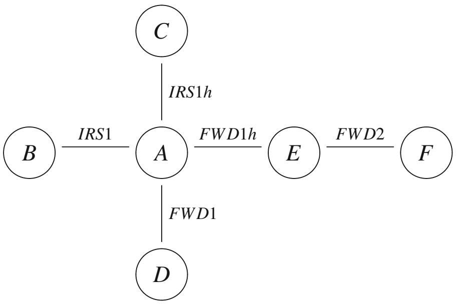
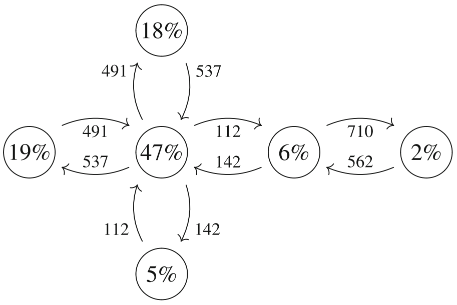


Fig. 4 Exposures: the nodes represent the same banks as in Fig. 3, the arrows represent that risk is induced from the bank on the tail onto the bank on the head, the weights on the arrows quantify that risk, and the percentages in the nodes represent the share of risk induced by that bank



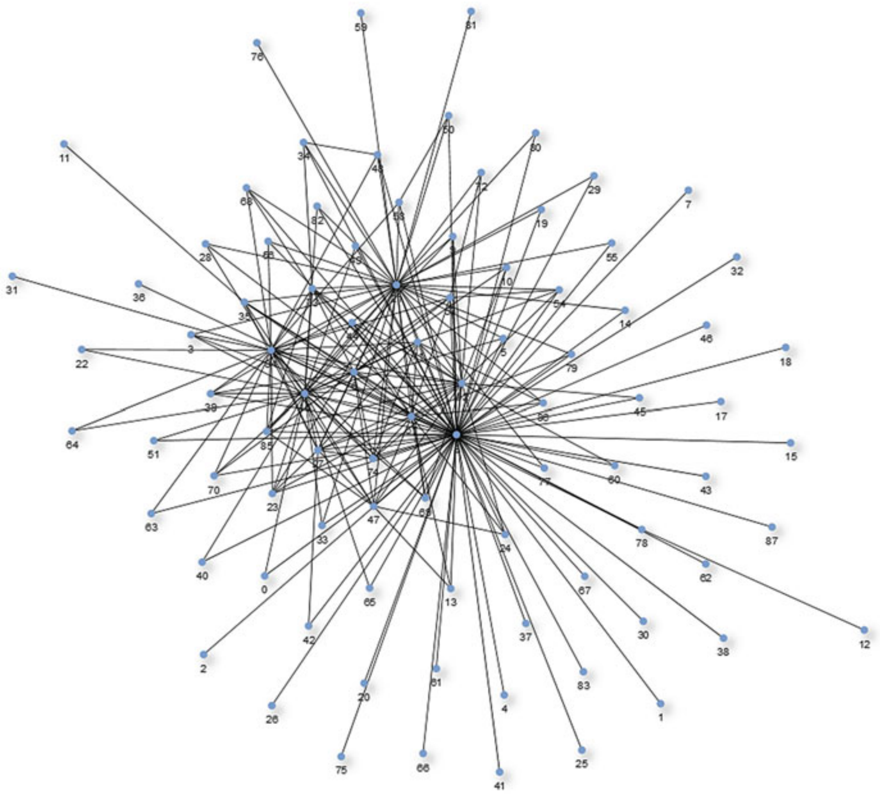


Fig. 5 A randomly generated trade relation graph

4. Systemic Risk Engine: In order to compute the individual risks in each banks' portfolio, we propose to use `ORE`, a novel open-source risk engine, see [19], which is based on `QuantLib` and implemented in `C++`. We extend this to a systemic risk engine, which computes not only the risk of any individual bank in the system but also aggregated systemic risk metrics, see component IV in Fig. 2. The main task is to automate entering all the generated financial systems into OREs `XML` configuration files, which is done using the Python library `lxml`.
5. Risk Graph Generation: Technically, a financial regulation can be implemented into the systemic risk engine as part of its input configuration, see component III of Fig. 2. Therefore, we can simulate the impact of a proposed financial regulation R_{prop} compared to a base regulation R_{base} by running the systemic risk engine on each of the financial systems $G_1, \ldots, G_s$ under regulation R_{base} as well as R_{prop}, see components IV and V of Fig. 2.
6. Classification, Data Mining, and Visualization: In a last step, we aggregate the generated data to allow hypothesis testing on whether or not the impact of R_{prop} over R_{base} has reduced systemic risk, see component VI of Fig. 2. In addition,

we use this raw data for data mining techniques such as anomaly detection to identify corner cases, in which R_{prop} has adverse effects or fails.

It seems obvious to bridge the gap between micro- and macro-prudential regulation by understanding the macro-prudential impact of a financial regulation via an aggregation of all the micro-prudential impacts on all banks, see again Fig. 1. The complexity of the system and of each of its banks as well as the large number of banks has made such an approach intractable for the traditional methods in financial economics. We believe that the current level of data science and risk management technology makes this bottom-up approach feasible for the first time. The approach has the advantage that it allows to understand the micro-prudential impact not only on one entity but on all of them simultaneously and systematically.

Notice that bridging a gap between a micro-perspective and a macro-perspective using data science to understand the macro-perspective as an aggregation of all micro-perspectives and organizing them in a graph model could in principle be applied to other domains as well, for instance payment systems, elections, or sentiment analysis.

2 A Graph Model of Financial Systems and Systemic Risk Metrics

A financial system can be modeled as a graph,[5] where the nodes represent the counterparties. Two counterparties are connected by an (undirected) edge if and only if they are conducting business with each other. The nodes as well as the edges can be enriched with additional data regarding the counterparties or the trade relations.

2.1 Trade Relation Graph

We capture this by the following formal definition, see Fig. 3.

Definition 1 (Financial System) A *financial system* $\text{FS} = (B, T, \tau)$ is an undirected graph $G = (B, T)$ called *trade relation graph* together with a *trade data function* τ, where:

- B is the set of nodes in the graph representing the banks.
- T is the set of (undirected) edges in the graph representing the trade relations between the banks. Two banks $b_1 \in B$ and $b_2 \in B$ are connected by an edge if and only if they have at least one trade with each other.

[5]See [4] for an overview of graph theoretic modeling of large-scale networks.

- $\tau : T \rightarrow Y$ is a *trade data function*, which to each trade relation t assigns additional trade relation data $\tau(t)$ (for instance a list of trade IDs) in some data space Y.

Optionally, this can be enriched to (B, T, β, τ), where:

- $\beta : B \rightarrow X$ is a *node data function*, which assigns each bank $b \in B$ additional data $\beta(b)$ (for instance its core capital) in some data space X.

In Fig. 3, we see an example of a financial system, where six banks (labeled A–F here) are trading bilaterally with each other in five trade relations. The trade data function τ assigns to each trade relation t the ID of a trade ($IRS1$ for instance stands for an Interest Rate Swap). While in this example, every trade relation only consists of one trade, banks often have many trades with each other.[6] While it is possible (both mathematically and technically) to assign more data than just a list of trade IDs to a trade relation, we stick to this simple trade data function for now. Optionally, one can supply this example with a node data function β, which is not shown in Fig. 3. A meaningful example of a function β is to assign to each bank b its core capital (or core capital ratio).

2.2 *Risk Graph*

Each trade in a trade relation imposes various types of risks (as well as rewards) on potentially both banks, which can be computed in various metrics by means of mathematical finance. By computing a fixed set of risk metrics for all trade relations in a trade relation graph, we obtain a graph that captures the risks between all the various banks in the system, see Fig. 4. Formally, we define this as follows:

Definition 2 (Risk Graph) Let $\mathrm{FS} = (B, T, \tau)$ be a financial system as in Definition 1 (optionally enriched by a node data function β). A weighted directed graph $\mathrm{RG} = (B, A, w)$ is called a *risk graph associated to* FS, where:

- B is the set of nodes in the graph representing the banks (i.e., FS and RG have identical nodes).
- A is the set of directed edges (often called arrows) in the graph RG. We add an arrow from a bank $b_1 \in B$ to a bank $b_2 \in B$ if and only if b_2 is exposed to some form of risk from b_1 as a consequence of their trades.
- $w : A \rightarrow \mathbb{R}^k$ is a multivariate weight function on the arrows quantifying the risks attached to each arrow measured in k metrics.

[6] A way to capture this technically is to give all trades in the financial system a globally unique ID. The trade relation function is then a function $\tau : T \rightarrow \mathscr{P}_f(\mathbb{N})$, where $\mathscr{P}_f(\mathbb{N})$ denotes the set of finite subsets of $\mathbb{N}$ (assuming each trade ID is a natural number). For instance, if the trades in Fig. 3 where enumerated by $0, 1, 2, \ldots, 4$ and IRS1 corresponds to 0, then for the trade relation t between bank A and B, we would have $\tau(t) = \{0\}$.

While the nodes in the trade relation graph and in the risk graph are the same, the edges are not. Each undirected edge in FS gives rise to one or two edges in RG. This is because there are trades like Interest Rate Swaps that induce risk from one counterparty to another and vice versa. In this case, we see two arrows between the counterparties that go in opposite directions, which in Fig. 4 is the case for all trade relations. Notice that there are also trades like FX Options where risk is induced in one direction only. On the arrows in Fig. 4, we only see one number, so $k = 1$, and the weight function chosen here is $w(a) := \mathrm{EEPE}(a) \in \mathbb{R}$, i.e., if $a = (b_1, b_2)$, then $w(a)$ denotes the EEPE that b_2 receives from b_1. Another example could be the PFE over a certain time horizon at a fixed quantile α (analogous to the US stress testing). Here, EEPE stands for *Effectivized Expected Positive Exposure* and PFE for *Potential Future Exposure*. These are standardized metrics to express risks in derivative portfolios, see [17, Sect. III] for a detailed discussion.

The weight function w in the risk graph associates a weight to the arrows. As banks often trade with many counterparties, we want to aggregate this information into a weight function on the nodes, i.e., the banks themselves. There is a purely graph theoretic procedure that extends a weight function on arrows to a weight function on nodes. As this method is less standard, we give the following definition.

Definition 3 (Weighted In/Out Degree) Let $G = (V, E, w)$ be a directed graph with nodes V, edges E, and a weight function $w : E \to \mathbb{R}^k$. In case $k = 1$, for each vertex $v \in V$, the quantities:

$$w^-(v) := \sum_{\substack{e \in E \\ e \text{ ends at } v}} w(e), \qquad w^+(v) := \sum_{\substack{e \in E \\ e \text{ starts at } v}} w(e) \tag{1}$$

are called the *weighted in/out degree*. Let $w(G) := \sum_{e \in E} w(e)$ be the total weight in the system. The quantities:

$$\rho^-(v) := \frac{w^-(v)}{w(G)}, \qquad \rho^+(v) := \frac{w^+(v)}{w(G)}, \tag{2}$$

are called *weighted relative in/out degree*. In case the weight function $w : E \to \mathbb{R}^k$ is multivariate, the weighted in/out degree $w : V \to \mathbb{R}^k$ and the relative weighted in/out degree $\rho : V \to \mathbb{R}^k$ are defined using Eqs. (1) and (2) in each component.

Intuitively, $w^\pm(v)$ is the total weight received respectively induced by v into the graph G. Notice that in case $k = 1$ and $w \equiv 1$, we have $w^\pm(v) = \deg^\pm(v)$, i.e., the weighted in/out degree is the ordinary in/out degree, which just counts the number of incoming respectively outgoing edges. Any of the quantities:

$$w(G), \qquad \max_{v \in V} w^+(v), \qquad \max_{v \in V} \rho^+(v) \tag{3}$$

are $\mathbb{R}^k$-real-valued metrics that capture the total amount of weight in the graph and its concentration.

2.3 Systemic Risk

Applying these graph theoretic concepts to systemic risk is straightforward.

Definition 4 (Systemic Risk) Let $\mathrm{RG} = (B, A, w)$ be a risk graph as in Definition 2. Extend the weight function $w : A \to \mathbb{R}^k$ on the arrows to a weight function $w^{\pm} : B \to \mathbb{R}^k$ using the weighted in/out degree from Eq. (1). We also define $\rho : B \to \mathbb{R}^k$ via Eq. (2). We regard any of the quantities in Eq. (3) as a *metric of systemic risk.*

As any metric that measures risk in a trade relation between two counterparties can be used as a weight function in a risk graph, we have produced a general procedure how to construct a systemic risk metric from that.

An example for such a metric is $w = \mathrm{EEPE}$. In that case for each bank b, the quantity $\mathrm{EEPE}^{+}(b)$ can be thought of as the cost of resolution if b defaults (like a loss given failure, but not a probability of failure). The quantities $\rho^{+}(b)$ are a metric that quantifies the concentration of these costs of resolution in the system.

3 Simulation Technology

In order to predict the impact of a financial regulation on systemic risk, we randomly generate financial systems as trade relation graphs, like the one shown in Fig. 3. For all netting sets containing all the trades in all the financial systems, we then compute risk metrics under different regulations. Using this data, we populate risk graphs like Fig. 4 for each regulation. Finally, we aggregate all these graphs to come to a conclusion on how a regulation impacts systemic risk. An overview of the technologies used in these various steps and their interactions is shown in Fig. 2. We will discuss the key challenges in this section.

3.1 Generation of Random Trade Relation Graphs

In order to generate random trade relation graphs like Fig. 3, various distributional choices have to be made.

3.1.1 Nodes

Generating the number of nodes is straightforward by choosing either a fixed value or drawing from a uniform distribution between an upper and lower bound.

3.1.2 Edges

How to randomly generate the edges such that the resulting trade relation graph forms a realistic financial system? Ideally, one should sample those from empirical data, but the trade relations in the real financial system are top secret. However, in [10] a statistical analysis of the macro-exposures in the Brazilian banking system has been performed using aggregated anonymized data provided by the Brazilian central bank. Due to this analysis, it is known that the degrees of the nodes in the trade relation graph follow approximately a Pareto distribution. Therefore, the problem can be split up into sampling a Pareto distributed sequence $\{d_1, \dots, d_n\}$, where n is the number of nodes in the graph, and second by creating a graph out of this sequence. While the first step is straightforward, the second is not.

Generating random graphs with a degree sequence prescribed from a given distribution is a hard problem in discrete mathematics. For Poisson distributed degree sequences, a solution has been developed by Erdős and Rényi in the 1960, see [12, 13]. With the advances of the WWW and social networks, generating random graphs with degree sequence sampled from an arbitrary given distribution became a useful tool. Therefore, substantial theoretical work as well as the search for efficient algorithms has been invested into this problem. We use the *erased configuration model* as implemented in the `Python` library `networkx` and described in [16]. Further details can also be found in [5, 8]. For random number generation, we use the `Python` library `numpy.random`.

3.1.3 Trades

Once a trade relation has been generated, the portfolio between the banks attached to this relation has to be populated with trades. For the purpose of this text, we keep it simple and choose a number k of trades, uniformly distributed between a fixed upper and lower bound, and then draw k times without replacement from a list of trade IDs. To generate the trades behind these IDs, we start with an FX Forward (EUR vs. USD) with a uniformly distributed maturity of up to 5 years, a uniformly distributed domestic amount between 100k and 100mn and a log-normally distributed strike rate, and also with an Interest Rate Swap (fixed vs. floating) with a uniformly distributed maturity of up to 10 years, a uniformly distributed notional between 100k and 100mn, and a uniformly distributed fixed rate between 0.01% and 5%. The choice between a Forward and the Swap as well as the sides of the trades are Bernoulli distributed with probability 0.5. We assume that all trades between two counterparties are in precisely one netting set.

3.2 The Open-Source Risk Engine (ORE)

Computing the exposure in even in just a single netting set of derivative trades is a hard problem in mathematical finance. The risk department of a bank invests substantial resources into developing and maintaining a risk management system capable of computing the banks exposure—in particular, if the bank chooses to use its own internal models. As those risk management systems are proprietary, they cannot be used for the purpose of this text.

Therefore, we use the *Open-Source Risk Engine (ORE)*, an open-source software, which performs risk management computations like a production system in a bank. It has been released by Quaternion Risk Management in 2015 as part of the Open-Source Risk initiative, see [19]. ORE models the risk in a derivative trade from the perspective of a single bank. Given the banks' portfolio, market data, and some additional configuration files, it projects the value of the trades into the future using a *Monte Carlo simulation* based on complex stochastic modeling of the risk factors. A detailed description of these risk factor models can be found in [14]. In case of a collateralized netting set, ORE also consumes parameters that specify the details of a netting agreement and simulates the value of collateral. This is a nontrivial matter. In particular, forecasting the IM is still subject to current research, see [3, 9, 11, 15].

3.3 Configuration and Aggregation

Each generated trade relation graph and each regulation has to be entered into OREs `XML` configuration before a run can be kicked off. This is realized via a `Python` script using `lxml`. After the run, OREs output consists of `csv` files containing all the risk metrics. We use `pandas` to extract those from the files and populate the risk graphs like Fig. 4. This puts us in a position to compute statistics on the risk metrics in the various graphs under various regulations.

4 Results: Impact of Collateralization on Systemic Risk

In this section, we apply the simulation technology described in Sect. 3 to study the impact of various collateralization regulations on systemic risk. The value of a derivative contract stems from future cashflows, which are not yet settled. If no collateral is exchanged, both parties are fully exposed to each other's credit risk (`REG_1`). To mitigate that risk, parties can exchange *Variation Margin (VM)* to cover the current exposure, i.e., the immediate loss suffered by the surviving counterparty on default (`REG_3`). This still does not reduce the exposure to zero as the surviving counterparty will need some time to close out the position. To mitigate the potential exposure caused by this gap, parties can exchange *Initial*

Margin (IM) on top of the Variation Margin (REG_4). A detailed description of these three regulatory regimes can be found in [17, Sect. II].

In [17, Sect. V], we presented a case study, which analyzes the impact of these three different regulatory regimes on the financial system with the structure depicted in Fig. 3. We found while increasing collateralization decreases systemic risk measured in EEPE (respectively, EEPE+, see Definition 3), i.e., the total cost of resolving a failed system decreases with the amount of collateral. However, collateralization has only little effect on the concentration of these costs (i.e., the associated $\rho+$ does not change much).

We now use the simulation technology described in Sect. 3.1 to generate 10 financial systems with 30–50 counterparties and trade relations, which are approximately Pareto distributed. This results in 2360 trades in total. These are faced from two sides, which yields 4720 technical trades in 1378 netting sets in ORE. We compute the EEPE under the regulatory regimes 1), 3), and 4) using 500 Monte Carlo paths. Regulatory regime 2) is omitted, because if one sets a global threshold, the result would depend on an arbitrary choice on whether or not the netting set exposures are distributed above or below that threshold, which is unrealistic. In Fig. 6, we can see that under regime 3), i.e., the full VM collateralization, the total EEPE reduction is at about 71%, and under regime 4), i.e., the VM & IM collateralization, the reduction is at about 93%. The average does not exhibit any outliers among the 10 financial systems analyzed, see Fig. 7.

On a macro-level, these numbers confirm unambiguously that the impact of any collateralization yields to a reduction in risk. As a byproduct of the simulation, we obtain exposure data of 1378 netting sets, which we can now mine to gain insight into all micro-impacts of the various regulations.

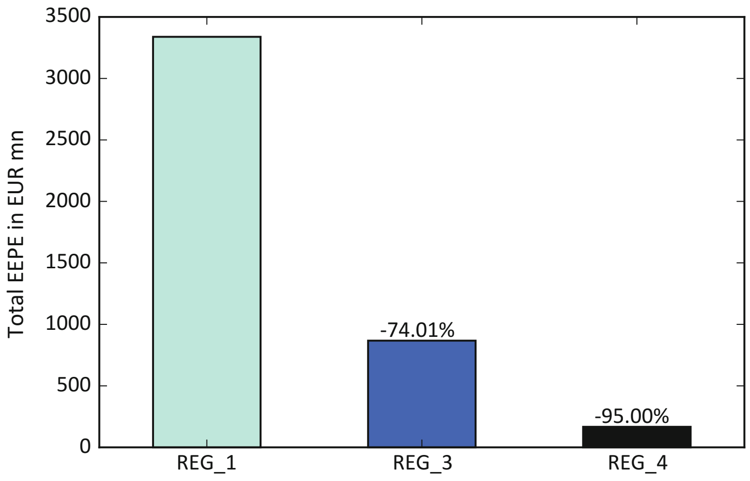


Fig. 6 Total reduction of EEPE

Fig. 7 Average relative reduction of total EEPE in the various financial systems

FS	VM	IM & VM
0	–78.00%	–97.22%
1	–74.63%	–94.72%
2	–75.97%	–96.86%
3	–71.47%	–94.45%
4	–70.23%	–93.16%
5	–72.89%	–94.45%
6	–73.88%	–94.16%
7	–75.46%	–96.31%
8	–74.44%	–95.04%
9	–71.43%	–91.95%

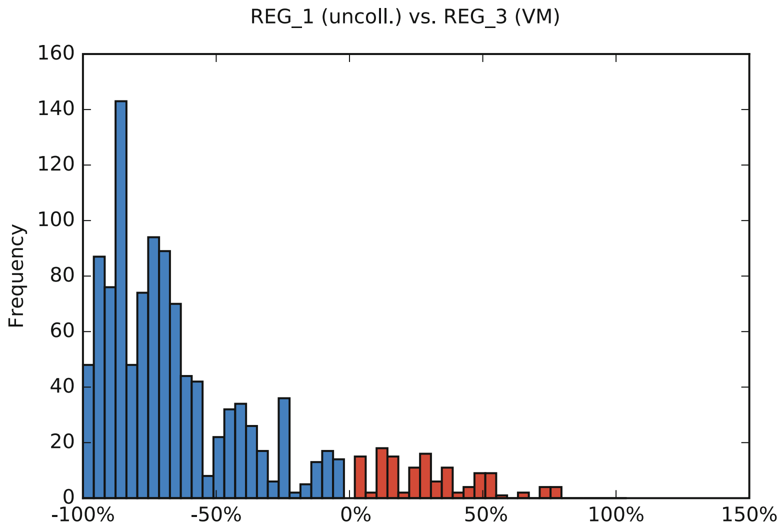


Fig. 8 Histogram of relative reduction in EEPE over all netting sets (197 out of 1378 have more than 150% increase and are not shown, 29 of those have zero uncollateralized EEPE). Mean:-57.42%, SD: 38.33%

In Fig. 8, we see the distribution of relative reductions in EEPE of the various netting sets when comparing `REG_1` (uncollateralized) with `REG_3` (VM collateralized). While most of the netting sets show a significant relative reduction in exposure, we can see that some of them also show a significant relative increase in exposure. The explanation for this is as follows: Assume that bank A has trades in a netting set with bank B. These trades are deeply out of the money for bank A, meaning that the markets have moved into bank B's favor. Then, the uncollateralized exposure for bank A is very low.[7] Under VM collateralization however, as the trades are deeply in the money for bank B, bank B will call bank A for variation margin.

[7]Due to the finite number of Monte Carlo paths, it is sometimes even numerically zero in the simulation.

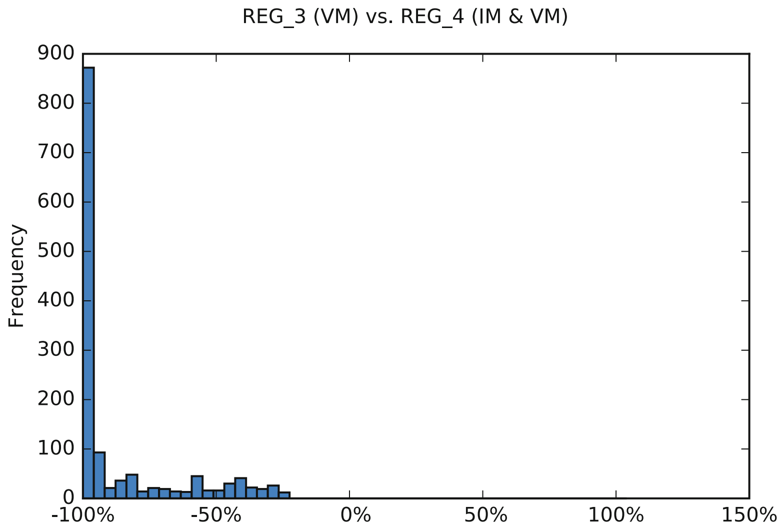


Fig. 9 Histogram of relative reduction in EEPE over all netting sets (0 out of 1378 have more than 150% increase and are not shown, 0 of those have zero VM collateralized EEPE). Mean: −85.78%, SD: 20.76%

Bank *A* will then pay the variation margin to bank *B*, where it is exposed to the default risk of *B*, because *B* might rehypothecate this variation margin. In some situations, this is resulting in higher exposure under VM collateralization than under no collateralization. We see that on a micro-level, VM collateralization can have an adverse effect in rare cases of netting sets, which are deeply out of the money.

Initial Margin cannot be rehypothecated and therefore, posted Initial Margin is not treated as being at risk.[8] In Fig. 9, we see the relative reductions in EEPE of the various netting sets when comparing `REG_3` (VM collateralization) vs. `REG_4` (VM & IM collateralization). Here, we can see that the effect of the additional IM overcollateralization unambiguously reduces the exposure further.

When comparing `REG_1` (uncollateralized) vs. `REG_4` (VM & IM collateralization) directly, we can see in Fig. 10 that the reduction in exposure is larger and distributed more narrowly compared with just the VM collateralization, see Fig. 8. There are still some netting sets left, which show an increase which is due to posted variation margin. However, this increase is smaller as under `REG_3` as it is mitigated by the additional IM collateral.

[8]It should be highlighted that in our simulation we model the bilateral trading between various banks, where Initial Margin is posted into segregated accounts. Derivatives that are cleared through a central counterparty (CCP) or exchange traded derivatives (ETDs) are not in scope of this simulation.

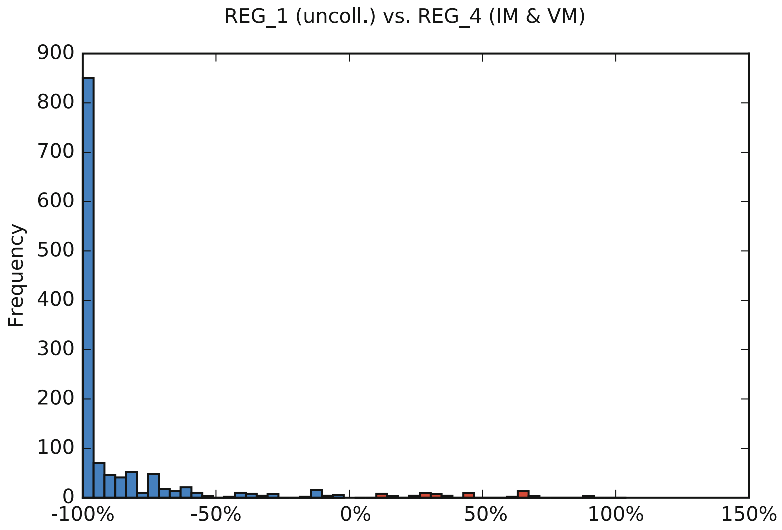


Fig. 10 Histogram of relative reduction in EEPE over all netting sets (69 out of 1378 have more than 150% increase and are not shown, 29 of those have zero uncollateralized EEPE). Mean: −83.83%, SD: 34.44%

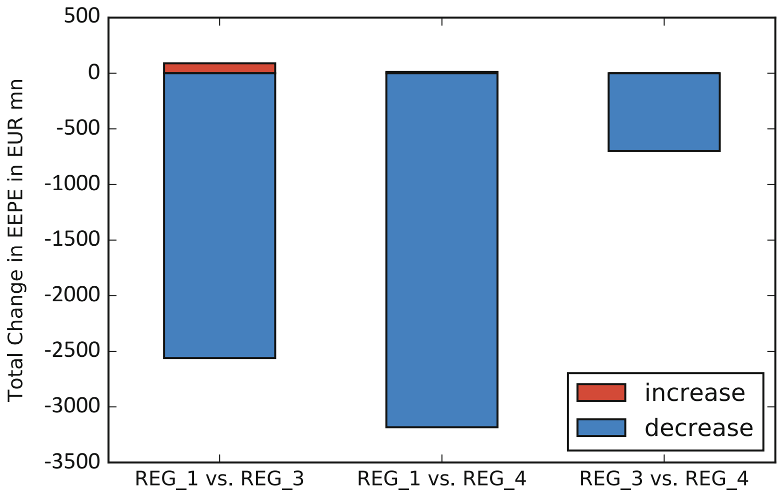


Fig. 11 Total increases and decreases in EEPE of all netting sets between the various regimes

It should be noted that while the increases in exposure we see in Figs. 8 and 10 are large in relative terms, they are actually quite small in absolute terms, see Fig. 11, where we compute the total increases and decreases in EEPE of all the netting sets separately.

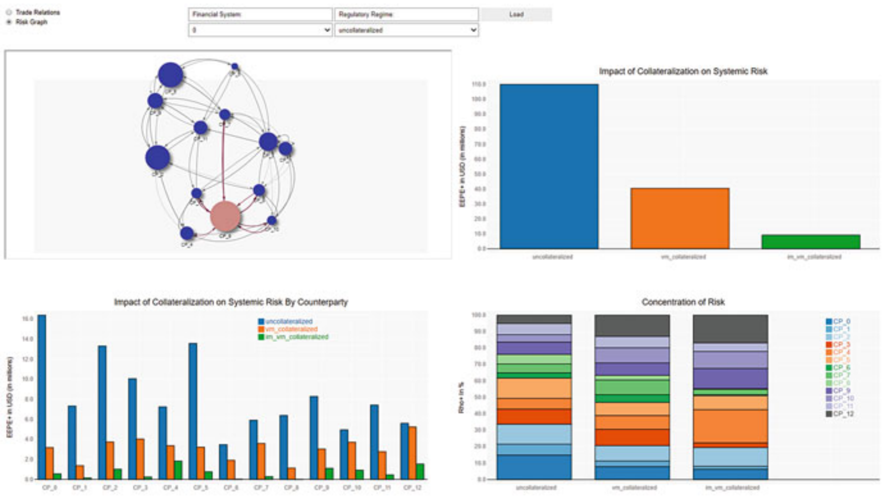


Fig. 12 Visualization of data outputted from systemic risk engine in Jupyter notebook dashboard. Depicts trade relations and risk graphs for various financial systems/regimes (top left) along with impact on risk within system (top right) and on individual counterparties measured in absolute terms (bottom left) and relative terms (bottom right). Can be configured and visualized with different regulations imposed

4.1 Jupyter Dashboard

To interactively explore the above results further, we developed a Jupyter dashboard,[9] see Fig. 12, using the `jupyter_dashboards` extension. Data on various financial systems and effects from various collateralization regulations outputted from the systemic risk engine is imported using `pandas`. The data is manipulated into proper forms for graphing purposes primarily using `pandas` and `numpy`. The trade relations and risk graphs are assembled using the *erased configuration model* from `networkx` and visualized using the `visJS2jupyter` module. The size of each node corresponds to the share of risk induced by the counterparty, while the width of the in-edges and out-edges quantifies the risk received respectively induced by the counterparty.

The remaining graphs are plotted using `bqplot`. The interactive menu is created using elements from the `ipywidgets` module that interact directly with elements within the notebook. The various financial systems and regulations can be chosen using dropdown menus with the option to switch between the trade relation and risk graphs. `Javascript` is employed to allow for dynamic execution of cells within the notebook and recreation of the graphs. Animations built into `bqplot` allow for seamless transitions across configurations, while the `networkx` graphs are regenerated upon each execution.

[9]Will be made available on http://fintech.datascience.columbia.edu/.

5 Synopsis

Our results can be summarized as follows:

1. Collateralization reduces the costs of resolution drastically,[10] see Fig. 6. In that sense, it reduces systemic risk.
2. The case study however suggests that collateralization does not change the distribution of these costs in the system, see [17, Sect. V].
3. The data mining exercise makes visible that in rare cases of netting sets, which are deeply out of the money, VM collateralization can actually cause an increase in exposure, which can be large in relative terms, but is small in absolute terms, see Figs. 8 and 11.

Notice that these results are an interplay of the aggregated macro-exposures and a systematic analysis of all micro-exposures, which would not be possible outside of the present framework.

Regarding practical applicability of this research, any bank b knows its own $\text{EEPE}^-(b)$ and has to report this quantity to the regulator. It is reasonable to assume that it also knows its $\text{EEPE}(b, b')$ for any other counterparty b' it does business with. (Especially for large complex banks, it is reasonable to assume that they can compute their own $\text{EEPE}^+(b)$ as well.) A regulator who would collect this information from all banks in the system could actually compute a graph like Fig. 4 representing the exposures in the real financial system as of today.

The research outlined in this chapter can be expanded into several directions.

Large-Scale Simulation The results from Sect. 4 we obtained by running the simulation technology described in Sect. 3 on a client desktop machine. We plan to deploy this on a cold environment and run a larger-scale simulation.

Capitalization and Central Clearing The $\text{EEPE}^+(b)$ of a bank b should be thought of as a measure of loss given failure, not a probability of failure. A bank with a large $\text{EEPE}^+(b)$ can actually still be very safe, if it has also large capital reserves. The key to take this into account is to add a capital adequacy ratio function $\text{CR} : B \to \mathbb{R}$ as a node data function in Definition 1 and to add RWA as an additional risk metric to the edges of the risk graph in Definition 2 (and then also to the nodes via the procedure described in Definition 4). The core capital of each bank can then be computed from that. This allows us to model a bank failure due to market risk, for instance by using the criterion:

$$\text{CC}(b) < \text{VaR}_q(b), \tag{4}$$

[10]The question to what extent exactly Initial Margin reduces exposure to CCR is subject to debate even when considering only one counterparty. In [2], the authors argue that when taking time lags between trade payments and margin reposting into account, the reduction is much smaller.

where VaR_q is the *Value at Risk* at some confidence level q. It also allows us to model a bank failure due to credit risk, for instance using the criterion:

$$\mathrm{CC}(b_1) < \mathrm{EEPE}((b_1, b_2)). \tag{5}$$

By increasing the confidence level q further and further and labeling a bank as defaulted if either Eqs. (4) or (5) is satisfied, one can study the collapse of the whole system. In particular, one can study how a default of one counterparty causes chain reaction of defaults. The values of q at which default events occur can be interpreted as p-values of the hypothesis that the system is safe. We believe this framework to be particularly suited to study the regulation around central clearing in a similar fashion.

Agents-Based Creation of Trade Relation Graphs Recall from Sect. 3.1 that we are currently generating the trade relations such that the degrees are approximately Pareto distributed, because this is in line with empirical data from [10]. If we prescribe the degrees, we obviously cannot get any insight into why those trade relations are formed. Therefore, we propose to enhance this by adding a trading strategy to the node data in Definition 1 for each bank. During the simulation, the banks would then trade based on that strategy and the evolution of the market and form those trade relations dynamically. It would be interesting to study under what conditions this produces Pareto distributed trade relations. As a byproduct, this enables studying the systemic market risks in trading strategies. What happens for example if everybody is trend following?

Derivative Market vs. Money Market In the current example, only the derivative business is considered. However, there is also an inherent credit risk in the classical money market. The financial regulation on collateralization has a significant impact on the interplay between the two: Unlike Variation Margin, the Initial Margin cannot usually be rehypothecated (that means you cannot reuse the Initial Margin you receive from one counterparty to post your Initial Margin to another). Raising the necessary funds to post Initial Margin will increase the business in the money markets and it would be interesting to test the hypothesis that this increases the systemic credit risk in the money market in a similar fashion. In that case the trade-off between the reduction of credit risk in the derivative market and the increase of credit risk in the money market should be further examined. Imagine a situation where a counterparty A hedges a trade it has sold to counterparty B by buying a reversed trade from counterparty C. Although A has zero market risk, it has to post Initial Margin twice, namely to B and C. If A borrows this Initial Margin directly or indirectly from B or C, the reduction in overall credit risk across both markets will be a lot lower.

Initial Margin and Funding Costs The reduction in EEPE in the derivative business comes at the cost of funding the collateral, in particular the Initial Margin. It is to be expected that these costs will be significant. On the other hand, the reduction in EEPE yields to a reduction in RWA, which yields to a reduction in the cost of capital. Therefore, it would be interesting to study the trade-off between the reduction in the cost of capital and the increase in funding costs. There are already various standardized metrics, called XVAs, to measure funding costs. In particular, the *Margin Value Adjustment (MVA)*, which captures the additional funding costs due to Initial Margin, contributes significantly. Studying the levels, evolution, and distribution of Variation Margin and Initial Margin and the associated XVAs as part of a case study as well as systemic versions of that would enable a quantitative evaluation of the trade-off.

Credit Risk vs. Liquidity Risk vs. Market Risk vs. Model Risk While the reduction of credit risk in Fig. 6 is very convincing, one should keep in mind that EEPE, the metric we use to quantify systemic risk, is a metric that captures Counterparty Credit Risk only. However, this is not the only source of risk in a financial system. Therefore, it would be more precise to say that collateralization reduces systemic credit risk. It is very reasonable to assume that pulling large amounts of money out of a financial system due to Initial Margin increases the liquidity risk in that system. Therefore, testing the hypothesis that collateralization increases systemic liquidity risk in a similar fashion and again study the trade-off between the two would be enlightening. We believe that the key to understand the link between credit risk and liquidity risk lies in a detailed analysis of the XVAs. While quantifying liquidity risk is difficult, it should be pointed out that value at risk (VaR), a standard metric to measure market risk—the most obvious source of risk—has a systemic analogue, namely CoVaR, see [1]. Finally, model risk can be added as well. This would allow us to study if for instance the increasing standardization of models increases or decreases the risk of system-wide model failures. What happens if everybody uses the same model and there exists a market condition under which it fails at one bank? It would be prudent to have a comprehensive simulation that studies the impact of collateralization on all these sources of risk and systemic risk.

References

1. Adrian, T., Brunnermeier, M.K.: CoVaR. Am. Econ. Rev. **106**(7), 1705–1741 (2016)
2. Andersen, L., Pykhtin, M., Sokol, A.: Does Initial Marign Eliminate Counterparty Risk?, risk.net (May 2017)
3. Anfuso, F., Aziz, D., Giltinan, P., Loukopoulos, K.: A Sound Modelling and Backtesting Framework for Forecasting Initial Margin Requirements, SSRN Pre-print (2016)
4. Bales, M., Johnson, S.: Graph theoretic modeling of large-scale semantic networks. J. Biomed. Inform. **39**(4), 451–464 (2006)
5. Bayati, M., Kim, J.-H., Saberi, A.: A sequential algorithm for generating random graphs. Algorithmica **58**(4), 860–910 (2010)
6. Billio, M., Getmansky, M., Lo, A.W., Pelizzon, L.: Econometric Measures of Systemic Risk in the Finance and Insurance Sectors. NBER Working Paper 16223, NBER (2010)
7. Bisias, D., Flood, M., Lo, A.W., Valavanis, S.: A Survey of Systemic Risk Analytics. Office of Financial Research, Working Paper #0001 (January 5, 2012)
8. Britton, T., Deijfen, M., Martin-Löf, A.: Generating simple random graphs with prescribed degree distribution. J. Stat. Phys. **124**(6), 1377–1397 (2006)

9. Caspers, P., Lichters, R.: Initial Margin Forecast–Bermudan Swaption Methodology and Case Study (February 27, 2018). Available at SSRN: https://ssrn.com/abstract=3132008
10. Cont, R., Moussa, A., Santos, E.: Network structure and systemic risk in banking systems. In Fouque, J., Langsam, J. (ed.): Handbook on Systemic Risk, pp. 327–368. Cambridge University Press, Cambridge (2013). https://doi.org/10.1017/CBO9781139151184.018
11. Caspers, P., Giltinan, P., Lichters, R., Nowaczyk, N.: Forecasting initial margin requirements–a model evaluation. J. Risk Manage. Financ. Inst. **10**(4), 365–394 (2017)
12. Erdős, P., Rényi, A.: On random graphs. Publ. Math. **6**, 290–297 (1959)
13. Erdőos, P., Rényi, A. On the evolution of random graphs. Publ. Math. Inst. Hung. Acad. Sci. **5**, 17–61 (1960)
14. Lichters, R., Stamm, R., Gallagher, D.: Modern Derivatives Pricing and Credit Exposure Analysis: Theory and Practice of CVA and XVA Pricing, Exposure Simulation and Backtesting (Applied Quantitative Finance), Palgrave Macmillan, Basingstoke (2015)
15. McWalter, T., Kienitz, J., Nowaczyk, N., Rudd, R., Acar, S.K.: Dynamic Initial Margin Estimation Based on Quantiles of Johnson Distributions, Working Paper (2018) Available at SSRN: https://ssrn.com/abstract=3147811
16. Newman, M.E.J.: The structure and function of complex networks. SIAM Rev. **45**(2), 167–256 (2003)
17. O'Halloran, S., Nowaczyk, N., Gallagher, D.: Big data and graph theoretic models: simulating the impact of collateralization on a financial system. Proceedings of the 2017 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2017, ASONAM '17, pp. 1056–1064. (2017)
18. O'Halloran, S., Nowaczyk, N., Gallagher, D.: Big data and graph theoretic models: simulating the impact of collateralization on a financial system. In Proceedings of the 2017 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2017, ASONAM '17, pp. 1056–1064. ACM, New York (2017). https://doi.org/10.1145/3110025.3120989
19. Open Source Risk Engine. www.opensourcerisk.org. First release in October 2016

Mining Actionable Information from Security Forums: The Case of Malicious IP Addresses

Joobin Gharibshah, Tai Ching Li, Andre Castro, Konstantinos Pelechrinis, Evangelos E. Papalexakis, and Michalis Faloutsos

Abstract The goal of this work is to systematically extract information from hacker forums, whose information would be in general described as unstructured: the text of a post is not necessarily following any writing rules. By contrast, many security initiatives and commercial entities are harnessing the readily public information, but they seem to focus on structured sources of information. Here, we focus on the problem of identifying malicious IP addresses, among the IP addresses which are reported in the forums. We develop a method to automate the identification of malicious IP addresses with the design goal of being independent of external sources. A key novelty is that we use a matrix decomposition method to extract latent features of the behavioral information of the users, which we combine with textual information from the related posts. A key design feature of our technique is that it can be readily applied to different language forums, since it does not require a sophisticated natural language processing approach. In particular, our solution only needs a small number of keywords in the new language plus the user's behavior captured by specific features. We also develop a tool to automate the data collection from security forums. Using our tool, we collect approximately 600K posts from three different forums. Our method exhibits high classification accuracy, while the precision of identifying malicious IP in post is greater than 88% in all three forums. We argue that our method can provide significantly more information: we find up to three times more potentially malicious IP address compared to the reference blacklist VirusTotal. As the cyber-wars are becoming more intense, having early accesses to useful information becomes more imperative to remove the hackers first-move advantage, and our work is a solid step towards this direction.

J. Gharibshah (✉) · T. Ching Li · A. Castro · E. E. Papalexakis · M. Faloutsos
University of California Riverside, Riverside, CA, USA
e-mail: jghar002@ucr.edu; tli010@ucr.edu; acast050@ucr.edu; epapalex@ucr.edu; michalis@ucr.edu

K. Pelechrinis
School of Information Sciences, University of Pittsburgh, Pittsburgh, PA, USA
e-mail: kpele@pitt.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_9

Keywords Security · Online communities mining · Forums

1 Introduction

How can we take the first-mover advantage away from hackers? We argue that hacker forums provide information earlier than other sources, and we should leverage these forums in our security intelligence. Here, we focus on a specific question. In particular, we want to extract as much useful information from hacker/security forums as possible in order to perform (possibly early) detection of malicious IP addresses, e.g., prior to their appearance on blacklists. The latter can exhibit large delays in their update and hence, new ways for labeling malicious IP addresses are needed [10]. In this study we will use the term "hacker forums" to describe online forums with a focus on security and system administration. Interestingly, we can classify these forums into categories: (a) main stream forums, like Wilders Security, and (b) "fringe" forums, like Offensive Community, where we find users with names like *satan911*. Some of these forums have been known to have hackers boast of attacks they have mounted, or sell tools for malicious purposes (think rent-a-botnet). For example, in our dataset there is a post that mentions "I give you a second server to have your fun with. Multiple websites on this server. So let's see if anyone can actually bring down the server." Right after that the hacker posted the IP, username, and password for anyone to access the server. In fact, there is a *show-off* section in these forums for people to broadcast their hacking "skills."

The overarching goal of this work is to mine the unstructured, user-generated content in security forums. Specifically, we focus here on collecting malicious IP addresses, which are often reported at such forums. We use the term security forum to refer to discussion forum with a focus on security, system administration, and/or more generally, system-related discussions. The users in these forums include: security professionals, hobbyists, and hackers, who go on these forums to identify issues, discuss solutions, and in general exchange information.

Let us provide a few examples of how users report IP addresses, which may or may not be malicious. Posts could talk about a benign IP address, say in configuration files, as in the post:

> [T]his thing in my hosts file: 64.91.255.87 ... [is] it correct?

At the same time, posts could also report compromised or malicious IP addresses, as in the post:

> My browser homepage has been hijacked to http://69.50.191.51/2484/.

The challenge is to automatically distinguish between the two. By doing so, we can provide a new source of information of malicious IP addresses directly from the affected individuals. Formally, we can state the problem as follows:

Key Question: Malicious IP Detection Given a set of posts $\mathscr{P}_F$ that may contain IP addresses and users $\mathscr{U}_F$ of a security forum F, as well as, the features Φ_p, $\forall p \in$

Table 1 Extracting useful information; number of malicious IP addresses found by InferIP and not by VirusTotal

		IP found by	
Dataset	Total IP	Virus Total	InferIP only
Wilders Security	4338	216	**670**
Offensive Community	7850	339	**617**
Ashiyane	8121	133	**806**

$\mathscr{P}_F$ and Φ_u, $\forall u \in \mathscr{U}_F$ for the posts and the users, respectively, can we determine if a given IP address i is malicious or not?

The set of features $\mathscr{P}_F$ includes attributes such as the text of the post, the posting user, the time of post, etc., while $\mathscr{U}_F$ includes information such as the date of a user joining the forum, the number of posts the user has made, etc. The above problem has two associated questions:

(a) **Exclusivity:** How many IP addresses can we find that are never reported by other reference sources?
(b) **Early warning:** How much earlier are malicious IP addresses reported in a forum compared to reference sources, for the IP addresses reported by both?

Most previous studies in this area have focused on structured information sources, such as security reports, or malware databases. In fact, many efforts focus on addressing security problems using knowledge obtained from the web, as well as social and information networks. These efforts are mainly focused on analyzing structured sources (e.g., [11]). However, studies assessing the usefulness of (unstructured) information in online forums have only recently emerged (e.g., [19]). These studies are rather exploratory and provide evidence of the usefulness of the data in the forums, but do not provide a systematic methodology or ready-to-use tools, which is the goal of our work. We discuss existing literature in more detail later in Sect. 5.

The motivation of our work is to provide more information to security analysts and systems. We want to enhance and complement, but not replace, existing efforts for detecting malicious IP addresses. For instance, many IP blacklists enlist an IP as malicious after a number of reports above a pre-defined threshold have been made for the specific address. Depending on the threshold and the reactivity of the affected users/systems, this might take several days, weeks, or months. Therefore, a system, like the one proposed here, can identify and point to malicious IP address to blacklist services and firewalls.

We propose InferIP, a systematic approach for identifying malicious IP addresses among the IP addresses, which are mentioned in security forums. A key novelty is that we use the behavioral information of the users, in addition to the textual information from the related posts. Specifically, we customize and use a sparse matrix regression method on this expanded set of features. By design, our framework is applicable to forums in different languages as it relies only on the behavioral patterns of users and simple word counts, and not a complex language-specific natural language processing technique. From a technical point of view the challenge in designing a solution to our key question is most IP addresses mentioned in these forums are not malicious. We show that our system can add a significant number of previously unreported IP address to existing blacklist services. Finally, as an engineering contribution, we develop a customizable tool to facilitate the crawling of forums, which we discuss in the next section.

Our results can be summarized into the following points:

(a) **Our method exhibits precision and recall greater than 88% and 85%, respectively, and an accuracy over *malicious* class above 86%** in the 10-fold cross validation tests we conducted for the three different forums. In partially answering our key question, if our method labels a currently non-blacklisted IP as malicious, there is a high chance that it is malicious, given our high precision.
(b) **Our method identifies three times more malicious IP addresses** compared to VirusTotal [21], a widely used aggregator of 60 blacklists of IP addresses. Across our three forums, we find more than 2000 potential malicious IP addresses that were never reported by VirusTotal (Table 1).
(c) **Our method identifies more than half of the IP addresses at least 3 month earlier than VirusTotal**. We study the malicious IP addresses that are identified by both VirusTotal and InferIP. We find 53%, 71%, and 62% of these IP addresses in Wilders Security, Offensive Community, and Ashiyane, respectively, at least 3 months earlier than they were reported in VirusTotal.
(d) **The number of reported malicious IP addresses has increased by a factor 8 in 4 years.** We find that the number of malicious IP addresses has increased from roughly 100 in 2011 and 2012 to more than 800 in 2016. This could be attributed to either an increase in the user base, an increase in the number of attacks, or a combination of the two.

2 Data Collection and Basic Properties

We have collected data from three different forums relevant to our study: (1) Wilders Security [22], (2) Offensive Community [15], and (3) Ashiyane [3]. The first two forums are mainly written in English, while the last forum is an Iranian forum, in Farsi.[1]

Our Data Collection Tool We develop a customizable universal tool to make the crawling forums easier. The challenge here is that each forum has its own format and layout. Our tool requires only a custom configuration file, before crawling a new forum. In configuration file, we specify entities in the forum which are needed such as user ID, post's date, post's content, etc., by XML path language known as *Xpath*. Leveraging our current configuration files, the task of crawling a new forum

[1]Our software and datasets will be made available at http://www.hackerchatter.org/.

Table 2 The collected forums

Forum	Threads	Posts	Users	Active days
Wilders Security	28,661	302,710	14,836	5227
Offensive Community	3542	25,538	5549	1508
Ashiyane	67,004	279,309	22,698	4978

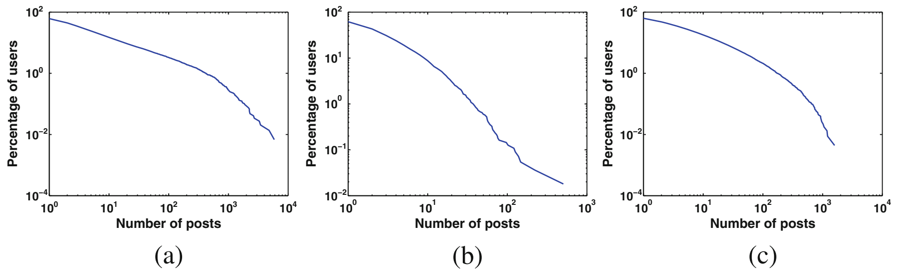


Fig. 1 CCDF of the number of posts per user (log–log scale). (**a**) Wilders Security. (**b**) Offensive Community (**c**) Ashiyane

is simplified significantly. Using our crawler, we collect data from three forums, two English and one in Farsi, for a total number of more than 30K users and 600K posts.

We use VirusTotal [21] as our reference blacklist IP addresses, since it is an aggregator, and combines the information from over 70 other blacklists and resources. VirusTotal is free to end users for non-commercial use and is a private API to query the services in the rate of more than 4000 IP addresses per minute. It is provided upon requests for academic purposes.

We provide some basic statistics for our three forums in Table 2. Offensive Community and Ashiyane are two fringe forums in different languages. In these forums there is a section where people openly boast about their achievement in hacking. They share their ideas and *tutorials* on how to break into vulnerable networks. On the other hand, Wilders Security as a mainstream forum is mostly used to protect non-experts against attacks such as browser hijacking, and provide solutions for their security problems.

For completeness, we present some of the terms we use here. A user is defined by a login name registered with the site. The term post refers to a single unit of content generated by a user. A thread refers to a collection of posts that are replies to a given initiating post.

In Figs. 1 and 2, we present the cumulative complementary distribution function of the number of posts per user and the number of threads per user, respectively. As we can see in all the cases the distributions are skewed, that is, most of the users contribute few posts in the forums and engage with few threads. In Wilders Security, 85% of users post less than 10 posts each, while 5.2% of the users post more than 50 posts. We find that 70% of the users post in only one thread and only 8% of the users are active in more than 10 threads. This skewed behavior is typical for online

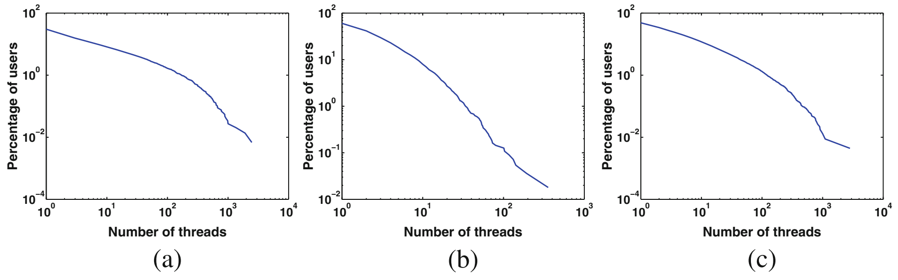


Fig. 2 CCDF of the number of thread per user (log–log scale). (**a**) Wilders Security. (**b**) Offensive Community (**c**) Ashiyane

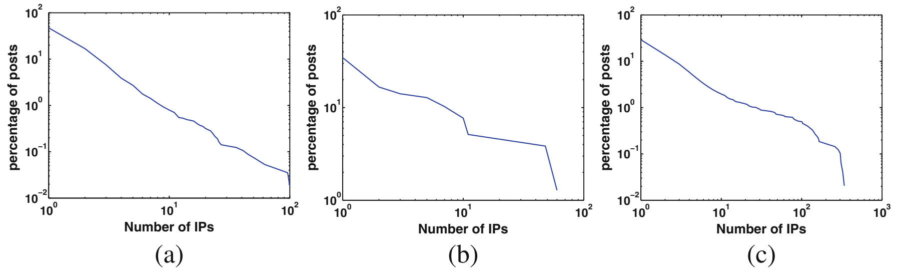


Fig. 3 CCDF of the number of IP addresses per post (log–log scale). (**a**) Wilders Security. (**b**) Offensive Community (**c**) Ashiyane

users and communities [7]. We will use features to capture aspects of both these user properties, as we will see in the next section.

In Fig. 3, we present the cumulative complementary distribution function of the number of IP addresses that appear in each post. The skewed distribution shows that most of the posts contain a few number of IP address. We find that 84.2% of the posts with IP addresses in Wilders Security and 84.1% in Offensive Community have two or less IP addresses. In Ashiyane, 87.2% of these posts contain less than two IP addresses. Interestingly, in Ashiyane, we find 1% of the IP containing posts with more than 100 IP addresses. We investigated and found that typically, these posts provide benign IP addresses of proxies' servers to fellow administrators.

Groundtruth for Training and Testing In order to build and evaluate our model we need to obtain a reasonably labeled dataset from IP addresses that appear in the posts of the security forums. For that, we use the VirusTotal service and assign malicious labels to an IP that has been reported by this service. The number of malicious IP addresses that we have used with the corresponding posts is shown in Table 1 as the IP found by VirusTotal. Note that the absence of a report on VirusTotal does not necessarily mean that the IP is benign. However, a listed IP address is most likely malicious, since VirusTotal as most blacklist sites requires a high threshold of confidence for blacklisting an address. This way, we find in total 688 malicious IP addresses for our forums as shown in Table 1.

Using this labeling process we have collected all the IP addresses that have appeared on our forums prior to their report on VirusTotal. For building our model, we also randomly select an equal number of IP addresses that have not been reported as malicious and via manual inspection further assess their status. Finally, for every security forum we have a different dataset and hence, we build a different model.

3 InferIP: Malicious IP Detection

We propose a method to identify whether an IP address within a post is malicious. For example, although many users report a malicious IP address, such as one that is attacking the user's network, there are also users who will mention a benign IP address when people discuss about network tutorials like setting up *Putty* or initiating a *SSH* connection.

While this task is simple for a human, it is non-trivial to automate. Adding to the challenge, different communities use different terminology and even different languages altogether (English and Farsi in our case). In order to overcome these challenges, we use a diverse set of features and build a model to identify IP addresses that are potentially malicious.

Our approach consists of four steps that each hide non-trivial novelties:

Step 1: We consider the user behavior and extract features that profile users who post IP-reporting posts.

Step 2: We extract keywords from the posts and use information gain to identify the 100 most informative features.

Step 3: We identify meaningful *latent feature sets* using an unsupervised co-clustering approach [16].

Step 4: We train a classifier using these *latent feature sets* using 10-fold cross validation.

We describe each step in more detail.

Step 1: Behavioral Features We associate each user of the forum with a set of 11 features that capture their behavior. In particular:

- Number of posts: The total number of posts made by the user.
- Number of threads: The total number of threads the user has contributed to.
- Number of threads initiated: The total number of threads initiated by the user.
- Average thread entropy: The average entropy of the user distribution of the threads in which the user has contributed to.
- Number of active days: The number of days that the user generates at least one post.
- Average day entropy: The average entropy of the user distribution of the posts made on the days that the user is active.

- Active lifetime: The number of days between the first and the last post of the user.
- Wait time: The number of days passed between the day the user joined the forum and the day the user contributed their first post.
- Average post length: The average number of characters in the user's posts.
- Median post length: The median number of characters in the user's posts.
- Maximum post length: The number of character's in the user's longest post.

Step 2: Contextual Features Apart from the aforementioned behavioral features we also include features related with the context in which an IP address appears within a post. In particular, we consider the frequency of the words (except stop-words) in the posts. Words that are frequent only in few documents (posts in our case) are more informative than those that appear frequently on a larger corpus [18]. To this end, we use TFIDF to weight the various words/terms that appear in our data.

After calculating the frequency and the corresponding weights of each word in the dataset we end up with more than 10,000 features/terms. Hence, in the next step we select discriminative features by extracting latent features.

We begin by performing feature selection in order to identify the most informative features by applying the information gain framework [23]. Furthermore, in order to avoid overfitting we pick a random subset of posts from the whole dataset and select the highest ranked features based on *Information Gain* score. In this way, a subset of discriminative keywords, 100 in our model, are selected. It turns out that each user uses only a small number of those words, resulting in a sparse dataset which we wish to exploit in our model.

Step 3: Identifying Latent Feature Sets We also like to leverage latent similarities of different posts in some of the dimensions spanned by post features and behavioral features for the writer of the post. Essentially, we seek to identify groups of highly similar posts under a small number of features, which does not necessarily span the full set of features. The reason why we wish to pinpoint a subset of the features instead of the entire set is because this way we are able to detect subtle patterns that may go undetected if we require post similarity across all the features. We call those sets of features *latent feature sets*. To this end, we apply a soft co-clustering method, sparse matrix regression (SMR) [16], to exploit the sparsity and extract latent features of the post containing IP addresses. Given a matrix $\mathbf{X}$ of posts $\times$ features, its soft co-clustering via SMR can be posed as the following optimization problem:

$$\min_{\mathbf{a}_r \geq 0, \mathbf{b}_r \geq 0} \|\mathbf{X} - \sum_{r}^{R} \mathbf{a}_r \mathbf{b}_r^T \|_F^2 + \lambda \sum_{i,r} |\mathbf{a}_r(i)| + \lambda \sum_{j,r} |\mathbf{b}_r(j)|$$

where $\mathbf{a}_r$ and $\mathbf{b}_r$ are vectors that "describe" co-cluster r, which we explain below. Each $\mathbf{a}_r$ is a vector with as many dimensions as posts. Each value $\mathbf{a}_r(i)$ expresses whether post i is affiliated with co-cluster r. Similarly, $\mathbf{b}_r$ is a vector with as many

Table 3 Selecting a classifier: overall accuracy

Forum	Naive Bayes	3NN	Logistic regression
Wilders Security	91.9%	87.1 %	94.8%
Offensive Community	84.1%	83.2%	86.5%
Ashiyane	85.1%	82.3%	94%

dimensions as features, and $\mathbf{b}_r(j)$ expresses whether feature j is affiliated with co-cluster r. Parameter λ controls how sparse the co-cluster assignments are effectively controlling the co-cluster size. As we increase λ we get sparser results, hence cleaner co-clustering assignments. We tune λ via trial-and-error so that we obtain clean but non-empty co-clusters, and we select $\lambda = 0.01$ in our case.

Step 4: Training the Model We subsequently train a number of classifiers using the selected features based on **a** matrix. In particular, we examine (a) a Naive Bayes classifier, (b) a K-nearest neighbor classifier, and (c) a logistic regression classifier. Our 10-fold cross validation indicates that the Logistic regression classifier outperforms kNN and Naive Bayes, achieving high accuracy, precision, and recall (see Table 3).

Determining Feature Sets We investigate the effect of selecting different feature sets in classifying IP addresses in forums. To this end, we investigate three subsets of the features discussed earlier.

(a) **Words-Frequency** is the normalized frequency of the most informative words that appear in a post as discussed in Step 2.
(b) **Combined** is the set of features which consists of the combination of the words-frequency features, defined above, and user behavior features, which are extracted in Step 1. In other words, it is the union of the features in Steps 1 and 2.
(c) **Co-Clustered** is the latent set of features extracted in Step 3 by applying the co-clustering approach on the Combined features set.

We evaluate these three sets of features on their ability to enable the classification. In more detail, we use these features with a classifier to assess their effectiveness by computing the accuracy of the classifier to identify malicious IP addresses. According to the results which are shown in Fig. 4, the Co-Clustered features set exhibits higher accuracy by 4.1% compared to Words-Frequency. On the other hand, although the Combined features do not increase the accuracy compared to the Words-Frequency, the co-clustering method does. It extracts the latent features from the Combined features set and outperforms Words-Frequency and Combined in identifying malicious IP addresses.

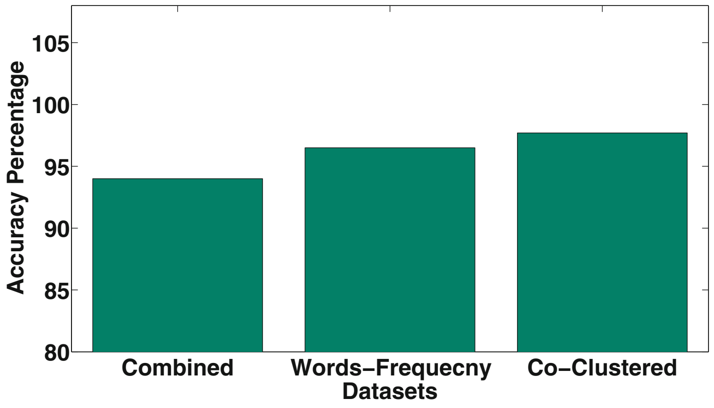


Fig. 4 Accuracy of different feature sets in Wilders Security forum to detect malicious IP

Table 4 InferIP evaluation: 10-fold cross validation evaluation (using logistic regression)

Forum	Instances	Precision	Recall	ROC Area
Wilders Security	362	0.9	0.94	0.96
Offensive Community	342	0.88	0.85	0.91
Ashiyane	446	0.9	0.92	0.92

3.1 Applying InferIP on the Forums

Having established the statistical confidence of our classifier, we apply it on the posts of the forums except the ones that we used in our ground truth. We use the logistic regression classifier as it exhibits the best performance (Table 4).

Applying InferIP on the forums shows that there is a wealth of information that we can extract from security forums in two aspects of the quantity and time of detecting malicious IP against VirusTotal.

(a) **Detecting more IP addresses.** With InferIP, we find an additional 670 malicious IP addresses in Wilders Security, 617 in Offensive Community, and 806 in Ashiyane (see Table 1). In other words, InferIP enables us to find three times additional malicious IP addresses in total compared to the IP addresses found on VirusTotal. It is interesting to observe that this factor varies among our three sites. For Ashiyane, our method finds roughly six times additional malicious IP addresses. With a precision of roughly around 90% and considering small amount of *False Positive* rate, our method can add a significant number of malicious IP addresses to a blacklist. Using the limited manual inspection, we confirm that the precision of the method on out-of-sample data is in the order of 88%.

(b) **Detecting malicious IP addresses earlier: more than half IPs, at least 3 months earlier.** Here we focus on the malicious IP addresses that are jointly identified by our method and VirusTotal and compare the time that they were

Table 5 Timely comparison between jointly detected malicious IP addresses in InferIP and VirusTotal. Reported percentage of malicious IP addresses which InferIP detected earlier than VirusTotal

	At least X months earlier		
Dataset	3	6	12
Wilders Security	53%	23%	14%
Offensive Community	71%	46%	21%
Ashiyane	62%	49%	37%
Average (across forums)	62%	39%	24%

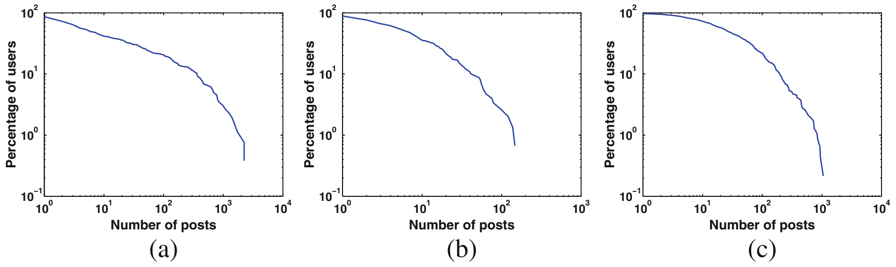


Fig. 5 CCDF of the number of overall posts per contributing user (who report malicious IPs) in log–log scale. (**a**) Wilders Security. (**b**) Offensive Community (**c**) Ashiyane

reported in each source, and show the results in Table 5 for 3, 6, and 12 months difference in time. We compare jointly detected IP addresses with InferIP and VirusTotal in terms of time that the IP addresses were mentioned in posts and the time they were reported on VirusTotal. We see that on average 62% of the malicious IP addresses with InferIP could be identified at least 3 months earlier than VirusTotal. We can see that with InferIP, we find 53%, 71%, and 62% of these IP addresses in Wilders Security, Offensive Community, and Ashiyane, respectively, at least 3 months earlier than in VirusTotal. We also identify 39% and 24% of the malicious IP addresses, respectively, at least 6 and 12 months earlier with InferIP.

Additional Stress-Testing of Our Accuracy In order to assess the performance of our approach, we randomly picked 10% of the labeled data with InferIP method and annotated them manually by human annotators. The calculated accuracy on the sampled data shows more than 85% accuracy on average over all datasets which is close but somewhat lower than the reported accuracy in Table 3.

Contributing Users Who are the users that report malicious IP addresses? We want to understand and ideally develop a profile for these users, which we will refer to as *Contributing* users. We start by considering the number of post these users post on the forums.

The Majority of IP Reporting Is Done by Highly Active (More Than 10 Posts Overall) in Ashiyane In Fig. 5, we show the cumulative complementary distribution function for the number of posts per *Contributing* user for Ashiyane.

More than 72% of the *Contributing* users post more than 10 posts overall, which we consider as high engagement given the distribution of posting that we saw in the previous section. Therefore, in Ashiyane, *Contributing* users are contributing significantly in reporting malicious IP addresses. Intrigued, we examined further and found that, among them, there are two users who have more than 1000 posts, 1058 and 2780 to be exact, and whose user-names are *"Classic"* and *"Crisis."* On the other side of the spectrum, 2.4% of *Contributing* users have posted a single post in the forum, and in that post they reported a malicious IP address.

The Majority of IP Reporting Is Done by Less Active Users (Less Than 10 Posts Overall) in Offensive Community In Fig. 5, we show the cumulative complementary distribution function for the number of posts per *Contributing* user for Offensive Community. Unlike Ashiyane, here 65% of the *Contributing* users have less than 10 posts overall. Going into more detail, roughly 12% of the *Contributing* users have a single post overall, while 26% of them have only two overall posts. The same behavior is observed in Wilders Security which is shown in Fig. 5.

Overall, there does not seem to be an obvious pattern between number of total posts and number of malicious IPs reported among *Contributing* users.

3.2 Case-Study: From Reported Malicious IPs to a DDoS Attack

We show that mining the forums could actually provide information about real events. We identify a link between a malicious IP address that our method detected with an actual DDoS attack.

We conducted the following analysis. We plot the time-series of the number of posts containing malicious IP addresses in Wilders Security from 2012 to 2013 found by InferIP. We show the time-series in Fig. 6. We observe some spikes on these time-series, which we further analyze. One of the spikes was in September 2012, and it reports a set of malicious IP addresses that were involved in an DDoS attack that month. That same thread continued being active, and in December of 2012, it was reported in that thread that attack was caused by *Nitol Botnet* due to a Microsoft's vulnerability [14].

We argue that this case-study points to additional layers of functionality that can be built upon our method and that can provide a semi-automated way to extract richer information beyond just reporting malicious IP addresses.

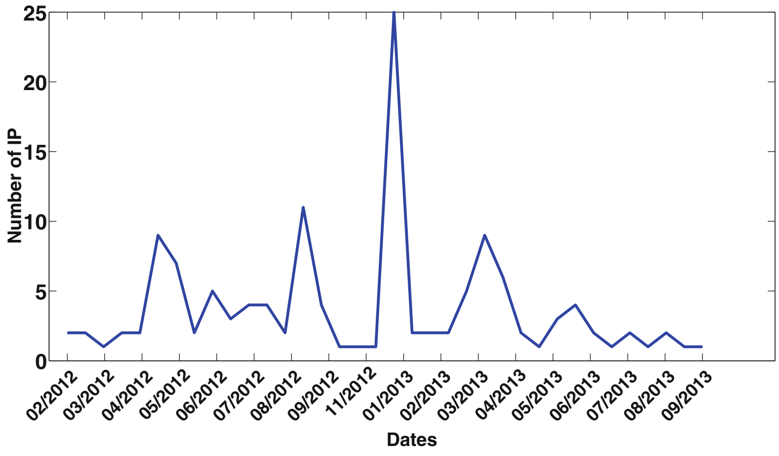


Fig. 6 Time-series of the number of posts containing malicious IP reported in each month for Wilders Security

3.3 Discussion and Limitations

Although our method exhibits pretty good accuracy overall, we attempt to understand its limitations and detect the source of misclassifications.

Limited Text in the Post The words in the post provide significant evidence for the classification. In some cases, some posts are very sparse in their text, which makes the classification of the included IP address harder. We consider these kinds of posts a significant contributor to misclassifications.

Characterization at the Post Level In our method, we classify an IP address by using features at the level of a post. Recall that roughly 86% of all posts across all forums has a single IP per post as shown in Fig. 3. In other words, having more than one IP address per post is already not very common. Furthermore, even more rarely, we have seen a few cases, where a post contains both a benign and a malicious IP address. As our method is currently set up, this will lead to errors in the classification. A straightforward solution is to consider examining the text surrounding each IP address within the post.

4 Spatiotemporal Analysis

In this section, we discuss the spatiotemporal features of the malicious IP addresses identified in security forums in Sect. 3.

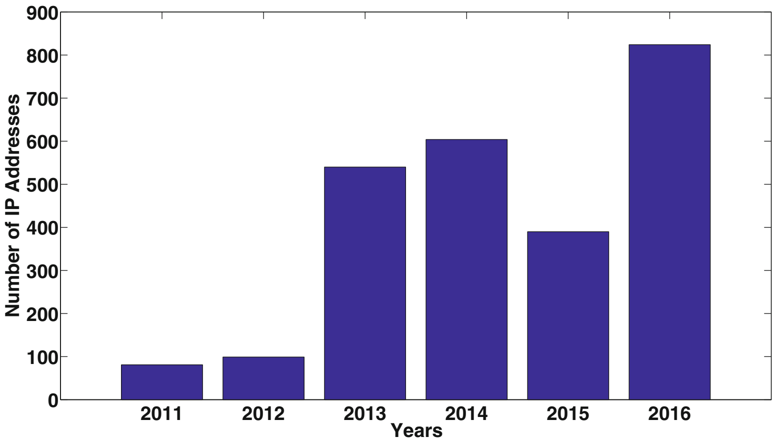


Fig. 7 Increasing trend: Malicious IP addresses reported on the forums each year

4.1 Temporal Analysis

The key question from a temporal point of view is if the number of reported malicious IP addresses increases or decreases over time.

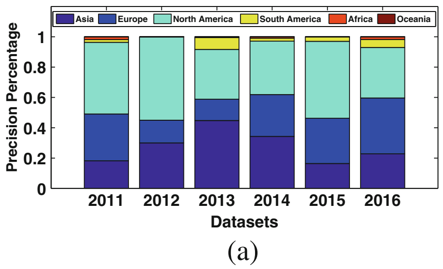
The Number of Reported Malicious IP Addresses Has Increased by a Factor 8 in 4 Four Years In Fig. 7, we plot the number of reported malicious IPs found by our method across all three forums between 2011 and 2016. We find that the number increased by a factor of 8: from roughly 100 to roughly 800. In spite of some decreases in years 2011, 2012, and 2015, it has a clear increasing trend.

4.2 Spatial Analysis

We study the geo-location of the identified IP addresses from Sect. 3. We utilize *GeoLite* database [9], which can show us the country and continent of an IP address. Here we focus on continents of the IP addresses location.

A natural question to ask is whether the geographical distribution of the malicious addresses differs between VirusTotal and InferIP. We investigate this in detail below.

VirusTotal: North America Hosts the Majority of the Reported Malicious IP Addresses We plot the percentage of the distribution of the IP addresses extracted from VirusTotal across continents in Fig. 8a between 2011 and 2016. We observe that the majority of the malicious IP addresses are located in the North America continent. There are two exceptions in 2013 and 2016 when Asia and Europe, respectively, contained most of the malicious IP addresses. Overall, Table 6 shows



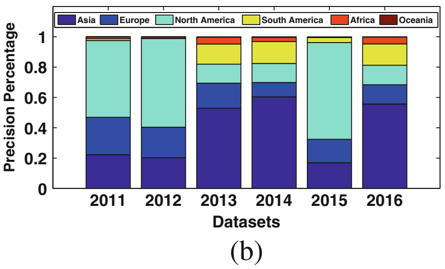


Fig. 8 Spatiotemporal distribution of malicious IP addresses detected by InferIP and VT. (**a**) VirusTotal. (**b**) InferIP

Table 6 Percentage of distribution of IP addresses across continents over all the years

	North America	Asia	Europe	South America	Africa	Oceania
InferIP	**46.7**	32.5	13.5	5.2	1.6	0.5
VirusTotal	**50**	26.5	20.4	2.4	0.6	0.17

the geographical distribution over all the years: North America, Asia, and Europe are the three most active continents in that order.

InferIP: North America Dominates Again, But South America and Africa Have Non-trivial Contributions We plot the percentage of the distribution of the IP addresses extracted form InferIP across continents in Fig. 8b between 2011 and 2016. We observe that North America hosts the majority of the reported malicious IP addresses again, but we find a more diverse global activity compared to what we observed in VirusTotal. For example, we can see that in years 2013, 2014, and 2016: (a) Asia has the majority of the malicious IP addresses, and (b) South America and Africa have a considerable percentage of malicious IP addresses. However, when seen across all years, the geographical distributions of the IPs in InferIP and VirusTotal are quite similar: North America, Asia, and Europe have the majority of the malicious IPs detected by InferIP similar to those of VirusTotal. In Fig. 9, we plot the geographical distribution of malicious IPs per continent across all years and all forums for InferIP and VirusTotal, while the exact numbers are shown in Table 6. Qualitatively the distributions look relatively similar, especially in the order of significance of the continents, but at the same, we can see that South America and Africa have a larger percentage of IP addresses in InferIP compared to those in VirusTotal.

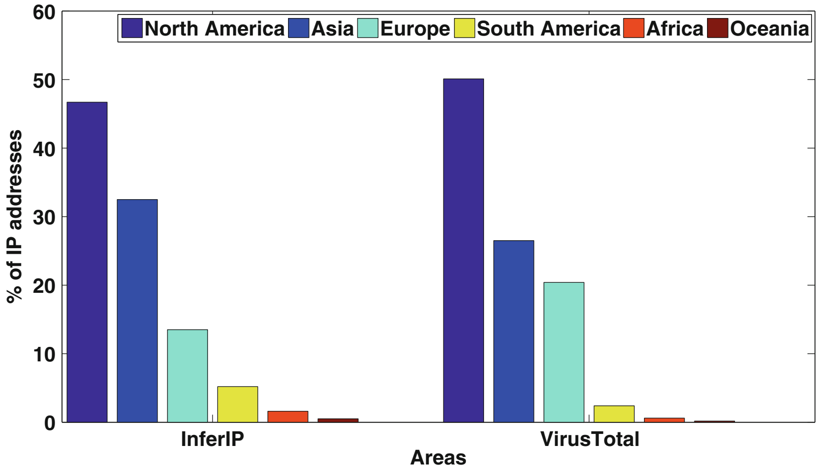


Fig. 9 The percentage distribution of malicious IP addresses in each continent across all three forums for InferIP and VirusTotal

5 Related Work

We briefly discuss three categories of relevant research:

(a) **Analyzing structured security sources.** There is a long line of research studying the ontology of cyber security and the automatic extraction of information from structured security documents. Iannacone et al. [11] developed a schema for extracting relevant concepts from various types of structured data sources. In another work, Blanco et al. [4] proposed methods to detect anomalies on the extracted ontology and network flow graph. Moreover, Bridges et al. [5] proposed a method to do entity labeling on structured data by utilizing neural networks. These works are complementary to ours as we focus on unstructured data, which poses different challenges.

(b) **Analyzing online security forums.** Recently security forums have been the focus of various studies that showcase the usefulness of the information present in security forums. For example, Motoyama et al. [13] present a comprehensive statistical analysis in underground forums. Others studies focus on the users' classification or the discovery of the relationships between the forum's members [1, 24]. Extracting different discussion topics in the forums and classifying the language of the codes posted in the forum has been done in [19]. Contrary to these studies, our work emphasizes on the development of automated systems that actually exploit the wealth of information in these security forums in order to enhance security. Similarly detecting malicious users on commenting platforms has been done on [12]. A recent work analyzes security forums to identify and geo-locate Canadian IP addresses focusing on spam and phishing

[8] and in another work, Portnoff et al. [17] studied the exchange of malicious services and tools and studied their prices on the security forums.

(c) **Analyzing blogs and social networks.** There have been a plethora of studies on blogs and social media, but their goals are typically not related to extracting security information [2, 6, 20]. The studies range from modeling user behavior [7] to inferring information about the user (demographics, preferences, and mental state), and to modeling the information propagation on online forums. Although interesting, the focus of these studies is significantly different from our goal here.

6 Conclusion

The take away message from our work is that there seems to be a wealth of useful information in security forums. The challenge is that the information is unstructured and we need novel methods to extract it. In this direction, a key insight of our work is that using behavioral and text-based features can provide promising results.

In support of this assertion, we develop a systematic method to extract malicious IP addresses reported in security forums. We utilize both behavioral, as well as textual features, and show that we can detect malicious IP addresses with high accuracy, precision, and recall. Our results in Table 1 are promising.

We then apply InferIP to all the posts we have collected. Although our classification is not perfect, our relatively high precision (hovering around 90% in Table 4) provides sufficient confidence in our results. We find three times as many additional malicious IP addresses as the original malicious IP addresses identified by VirusTotal. Furthermore, even for the jointly discovered IP addresses, at least 53% of the IP addresses were detected at least 3 months earlier than VirusTotal. The key message from our spatiotemporal analysis is that the number of reported malicious IP addresses is increasing overtime.

In the future, we plan to extend our work by extracting other types of security information. Our first goal is to detect malicious URLs mentioned in the forums. Our second and more ambitious goal is to identify the emergence of new malware, threats, and possibly attacks, which we expect to see associated with large numbers of panic-filled or help-requesting posts. Our final goal is to identify malicious users, since interestingly, some users seem to be promoting and selling hacking tools in these forums.

Acknowledgements This material is based upon work supported by an Adobe Data Science Research Faculty Award, and DHS ST Cyber Security (DDoSD) HSHQDC-14-R-B00017 grant. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the funding institutions.

References

1. Abbasi, A., Li, W., Benjamin, V., Hu, S., Chen, H.: Descriptive analytics: examining expert hackers in web forums. In: 2014 IEEE Joint Intelligence and Security Informatics Conference, pp. 56–63. IEEE, Piscataway (2014)
2. Althoff, T., Jindal, P., Leskovec, J.: Online actions with offline impact: how online social networks influence online and offline user behavior. In: Proceedings of the Tenth ACM International Conference on Web Search and Data Mining (WSDM'17), pp. 537–546. ACM, New York (2017)
3. Ashiyane. http://www.ashiyane.org/forums/
4. Blanco, C., Lasheras, J., Valencia-García, R., Fernández-Medina, E., Toval, A., Piattini, M.: A systematic review and comparison of security ontologies. In: 2008 Third International Conference on Availability, Reliability and Security, pp. 813–820. IEEE, Piscataway (2008)
5. Bridges, R.A., Jones, C.L., Iannacone, M.D., Testa, K.M., Goodall, J.R.: Automatic labeling for entity extraction in cyber security. arXiv preprint arXiv:1308.4941 (2013)
6. Cheng, J., Bernstein, M., Danescu-Niculescu-Mizil, C., Leskovec, J.: Anyone can become a troll: causes of trolling behavior in online discussions. In: Proceedings of the Conference on Computer-Supported Cooperative Work. Conference on Computer-Supported Cooperative Work, p. 1217. NIH Public Access (2017)
7. Devineni, P., Koutra, D., Faloutsos, M., Faloutsos, C.: If walls could talk: patterns and anomalies in Facebook wallposts. In: Proceedings of the 2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2015 (ASONAM'15), pp. 367–374. ACM, New York (2015)
8. Frank, R., Macdonald, M., Monk, B.: Location, location, location: mapping potential Canadian targets in online hacker discussion forums. In: 2016 European Intelligence and Security Informatics Conference (EISIC), pp. 16–23. IEEE, Piscataway, (2016)
9. Geolite. http://dev.maxmind.com/geoip/legacy/geolite/
10. Hang, H., Bashir, A., Faloutsos, M., Faloutsos, C. and Dumitras, T.: "Infect-me-not": a user-centric and site-centric study of web-based malware. In: IFIP Networking Conference (IFIP Networking) and Workshops, pp. 234–242. IEEE, Piscataway (2016)
11. Iannacone, M., Bohn, S., Nakamura, G., Gerth, J., Huffer, K., Bridges, R., Ferragut, E., Goodall, J. Developing an ontology for cyber security knowledge graphs. In: Proceedings of the 10th Annual Cyber and Information Security Research Conference (CISR'15), pp. 12:1–12:4. ACM, New York (2015)
12. Li, T.C., Gharibshah, J., Papalexakis, E.E., Faloutsos, M.: Trollspot: detecting misbehavior in commenting platforms. In: Proceedings of the 2017 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM '17), pp. 171–175. ACM, New York (2017)
13. Motoyama, M., McCoy, D., Levchenko, K., Savage, S., Voelker, G.M.: An analysis of underground forums. In: Proceedings of the 2011 ACM SIGCOMM Conference on Internet Measurement Conference (IMC'11), pp. 71–80. ACM, New York (2011)
14. Nitol-botnet. https://threatpost.com/tag/nitol-botnet/
15. Offensive Community. http://www.offensivecommunity.net
16. Papalexakis, E.E., Sidiropoulos, N.D., Bro, R.: From k-means to higher-way co-clustering: multilinear decomposition with sparse latent factors. IEEE Trans. Signal Process. **61**(2), 493–506 (2013)
17. Portnoff, R.S., Afroz, S., Durrett, G., Kummerfeld, J.K., Berg-Kirkpatrick, T., McCoy, D., Levchenko, K., Paxson, V.: Tools for automated analysis of cybercriminal markets. In: Proceedings of the 26th International Conference on World Wide Web, pp. 657–666. International World Wide Web Conferences Steering Committee
18. Ramos, J.: Using TF-IDF to determine word relevance in document queries. In: Proceedings of the First Instructional Conference on Machine Learning, vol. 242, pp. 133–142 (2003)

19. Samtani, S., Chinn, R., Chen, H.: Exploring hacker assets in underground forums. In: IEEE International Conference on Intelligence and Security Informatics (ISI), pp. 31–36. IEEE, Piscataway (2015)
20. Ugander, J., Karrer, B., Backstrom, L., Marlow, C.: The anatomy of the Facebook social graph. arXiv preprint arXiv:1111.4503 (2011)
21. Virustotal. http://www.virustotal.com
22. Wilders Security. http://www.wilderssecurity.com
23. Yang, Y., Pedersen, J.O.: A comparative study on feature selection in text categorization. In: Proceedings of the Fourteenth International Conference on Machine Learning (ICML'97), pp. 412–420. Morgan Kaufmann Publishers, San Francisco (1997)
24. Zhang, X., Tsang, A., Yue, W.T., Chau, M.: The classification of hackers by knowledge exchange behaviors. Inf. Syst. Front. **17**(6), 1239–1251 (2015)

Temporal Methods to Detect Content-Based Anomalies in Social Media

Jacek Skryzalin, Richard Field Jr., Andrew Fisher, and Travis Bauer

Abstract We develop a method for time-dependent topic tracking and meme trending in social media. Our objective is to identify time periods whose content differs significantly from normal, and we utilize two techniques to do so. The first is an information-theoretic analysis of the distributions of terms emitted during different periods of time. In the second, we cluster documents from each time period and analyze the tightness of each clustering. We also discuss a method of combining the scores created by each technique, and we provide ample empirical analysis of our methodology on various Twitter datasets.

1 Introduction

Social media platforms (Twitter, Facebook, etc.) allow users to instantaneously publish small, textual utterances. Taken individually, these utterances might have little content and provide little information. Taken in aggregate, however, they can provide insights into, for example, public health [6], political sentiment [23], and personality [7].

We develop a framework which allows us to detect and understand temporal anomalies in a collection of timestamped documents, such as those produced on social media. More explicitly, we identify time periods during which the produced documents' content differs drastically from the norm or shows unusually high focus

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC., a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525. SAND2017-12340 J.

J. Skryzalin · R. Field Jr. (✉) · A. Fisher · T. Bauer
Sandia National Laboratories, Albuquerque, NM, USA
e-mail: jskryza@sandia.gov; rvfield@sandia.gov; anfishe@sandia.gov; tlbauer@sandia.gov

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_10

or intensity, but we do not place further restrictions or specifications on the nature of the anomaly. As such, we focus on *unsupervised* techniques which allow the detection of an anomalous state without a prior specification of the exact nature of the anomaly.

We discuss related research and its relationship to the current work in Sect. 2. In Sect. 3, we discuss two methods of detecting anomalous behavior, as well as a way to fuse the results of these two approaches. In Sect. 4, we present empirical results of our methods on various Twitter datasets.

A preliminary version of this work was presented at the 2017 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining [18]. New contributions presented herein include more detailed discussions on cluster coherence and probabilistic fusion, a sensitivity analysis demonstrating robustness of the proposed method to data size, and an extension of our approach to handle nonstationary datasets, that is, datasets where term distributions can change over time.

2 Related Work

There has been a significant amount of research on the trends and dynamics of a corpus of timestamped documents. These methods have been used to study trends in a diverse collection of corpora, including those consisting of scientific papers [5, 8, 9], historical speeches [24], news stories [12, 25], and social media posts [11, 26]. Although there are numerous such techniques, they can be broken into roughly two categories.

The techniques of the first category are generally known as topic detection and tracking (TDT) algorithms. These algorithms attempt to incorporate temporal data into traditional topic modeling algorithms—algorithms whose primary purpose is to produce clusters of similar documents. Some of these algorithms use predefined categories (e.g., music news, sports news, and political news) and supervised learning techniques to classify each new document into one of the predefined categories [11]. Other techniques create vectors from each document and use traditional unsupervised clustering algorithms to produce custom categories [13, 19].

Still other TDT algorithms tackle topic detection and tracking using probabilistic Bayesian modeling. These algorithms are usually based loosely on latent Dirichlet allocation (LDA) [4]. LDA represents a topic as a distribution over words and considers each document to have been generated by sampling from a mixture of topics. Some temporally sensitive variants of LDA partition a corpus into time intervals, run LDA on each time interval, and connect the topic distributions from each time interval with the topic distributions of neighboring time intervals [1, 5]. Other temporally sensitive variants of LDA associate each topic distribution with a temporal distribution to encourage each topic to occur in a relatively concentrated time period [24].

The second category of trend-identifying algorithms consists of techniques which provide the user with a set of memes, defined as (clusters of) important words or phrases, and the periods of time where each meme is considered especially important. Memes may or may not need to be specified in advance by the user, and the importance of a meme is typically related to the frequency of mentions per time. Various novel approaches have been developed to measure the importance of a meme. Kleinberg et al. measure the importance of a meme by fitting an infinite automaton to the temporal distribution of mentions of that meme [10, 12]. He and Parker construct a physical model of importance using proxies for a meme's mass and velocity derived from the temporal distribution of mentions of a meme and the context in which the meme occurs [9]. Swan and Allan extract important terms from temporal slices of a corpus using a χ^2 significance test [21]. Shasha et al. deem a meme m important in a time window w if a user-specified function $f(m, w)$ is greater than a user-specified threshold, and they have constructed efficient data structures and algorithms for identifying such memes [27, 28].

Although the algorithms discussed in this section successfully track temporal aspects of topics and/or identify trending memes, they tend to focus more on the topics and content being tracked and less on the relative importance of different slices of time. In this work, we revisit the issue of topic tracking and meme trending with a temporal focus. Rather than analyzing topics themselves, we identify time periods with unusually high or anomalous trendiness. Moreover, our techniques satisfy two properties which allow them to function well with minimal prior configuration. First, our methods are completely unsupervised—the algorithms are able to function without specifying categories or memes to be tracked. Unlike the work of Pennebaker et al. [14, 22], which focuses on the temporal correlation of pronoun usage and mental state, we discover both anomalous time periods and interesting textual markers which provide insights into the nature of the anomaly. Second, our methods are largely independent of the arrival rate of documents; we assume that any data we see has been sampled from a larger distribution, and we would like our methods to be able to accommodate differing sample sizes and sampling rates.

3 Methods

In this section, we present two techniques for studying term and topic trends from the perspective of identifying anomalous time periods. The first technique focuses on the variation of term distributions and highlights time periods whose term distributions differ drastically from baseline. The second technique uses clustering to construct a rough metric for topic coherence, which we expect to be higher when an unusually large percentage of documents share a topic.

We assume that we have time periods $t_1, t_2, \ldots, t_r$ and associated corpora $C_{t_1}, \ldots, C_{t_r}$ of documents, where C_{t_i} consists of all documents produced during time period t_i. We also assume that we have a corpus C_0 which serves as a "baseline" for our term distribution analysis. In our experiments, we use as the baseline corpus C_0 the union $C_0 = C_{t_1} \cup \cdots \cup C_{t_r}$.

3.1 Term Distribution Analysis

Our first technique utilizes information-theoretic analyses of the distributions of terms seen across varying time periods. Our analysis begins with Zipf's law—the observation that the nth most common word in a corpus occurs with frequency proportional to $n^{-\alpha}$ for some $\alpha > 0$ [16]. The parameter α varies based on language and corpus type (research articles, Twitter posts, etc.), yet α is surprisingly constant across different corpora of the same type. However, the distribution of terms in a corpus can vary widely, and it is this variation that we analyze.

First, for each term w, we construct a probability $p(w)$ (resp., $q(w)$) associated with the term w and some corpus C_t (resp., C_0) via one of the following:

- Document frequency: $p(w)$ is the proportion of documents in C_t containing w.
- Term frequency: $p(w)$ is the proportion of all terms in C_t which are equal to w.
- Weighted term frequency: $p(w)$ is a document-weighted proportion of all terms in C_t constructed so that all documents are weighted equally, that is:

$$p(w) = \frac{1}{|C_t|} \sum_{d \in C_t} \frac{\text{number of words in } d \text{ equal to } w}{\text{number of words in } d}, \tag{1}$$

  where $|C_t|$ denotes the number of documents in C_t.

It will be made clear which of these choices we use for a given analysis by the ensuing discussion.

The values $\{p(w)\}_w$ form a distribution (i.e., they are nonnegative and sum to 1) when defined using the term frequency or weighted term frequency option, but not when defined using the document frequency option. The non-weighted and weighted term frequency definitions differ in that the term frequency option assigns equal weight to each term, whereas the weighted term frequency option assigns equal weight to each document.

The Kullback–Leibler divergence $\mathrm{KL}(p\|q)$ between $\{p(w)\}_w$ and $\{q(w)\}_w$ is defined as:

$$\mathrm{KL}(p\|q) = \sum_w p(w) \log\left(\frac{p(w)}{q(w)}\right). \tag{2}$$

The Kullback–Leibler divergence is an asymmetric measure of the difference between two probability distributions which measures the number of extra bits needed to encode p when using a coding scheme optimized for q rather than a coding scheme optimized for p.

Since the Kullback–Leibler divergence is asymmetric, it is common practice to use the Jensen–Shannon divergence, a symmetrized version of the Kullback–Leibler divergence, when constructing a distance metric on probability distributions. However, we define an antisymmetric version of the Kullback–Leibler divergence via:

$$\mathrm{AKL}(p\|q) = \mathrm{KL}(p\|q) - \mathrm{KL}(q\|p) = \sum_w (p(w) + q(w)) \log\left(\frac{p(w)}{q(w)}\right). \tag{3}$$

When analyzing the trends of a corpus C_t, we find it most useful to analyze the term-wise contributions to AKL $(p\|q)$. We thus define the pointwise antisymmetric Kullback–Leibler (PAKL) score of a term w to be

$$\mathrm{PAKL}_{(p\|q)}(w) = (p(w) + q(w)) \log\left(\frac{p(w)}{q(w)}\right). \tag{4}$$

The value $\mathrm{PAKL}_{(p\|q)}(w)$ satisfies the following properties:

1. $\mathrm{PAKL}_{(p\|q)}(w)$ is positive if $p(w) > q(w)$ and is negative if $p(w) < q(w)$.
2. $\mathrm{PAKL}_{(p\|q)}(w)$ approaches zero as $p(w)$ approaches $q(w)$.
3. $\left|\mathrm{PAKL}_{(p\|q)}(w)\right|$ increases as either (a) $p(w)$ stays constant and $q(w)$ approaches 0, or (b) $q(w)$ stays constant and $p(w)$ approaches 0.

Thus, when analyzing a set of timestamped corpora, we can monitor the time evolution of PAKL scores to determine whether the relative frequency of a term is increasing, decreasing, or staying constant in time. We can also sum all PAKL scores, all positive (resp., negative) PAKL scores, or the most n positive (resp., negative) PAKL scores in each corpus C_t in order to construct a score which measures the relative trendiness (or, in the case when relative common words experience a drop in usage, anti-trendiness) exuded by C_t.

We note that the third property listed above is key to our analysis. If we had instead chosen to analyze a "pointwise" version of the Kullback–Leibler divergence, we might have defined a score $\mathrm{PKL}_{(p\|q)}(w)$ via:

$$\mathrm{PKL}_{(p\|q)}(w) = p(w) \log\left(\frac{p(w)}{q(w)}\right). \tag{5}$$

Note, however, that $\mathrm{PKL}_{(p\|q)}(w)$ is unable to differentiate between terms w such that $p(w) \approx q(w)$ and terms w such that $p(w) \approx 0$, since in both cases, $\mathrm{PKL}_{(p\|q)}(w) \approx 0$.

3.2 Cluster Coherence

Our second topic-based approach to the temporal analysis of a series of corpora is based on the idea that we can construct tighter clusters of documents during a time period when there is a heightened focus on a relatively small set of concepts. The procedure for this technique is as follows:

1. Obtain (GloVe) word vectors for the data.
2. Using the word vectors, derive a set of "corpus vectors" to represent the data in C_t.
3. Cluster the corpus vectors.
4. Obtain scores from the clustering which measure cluster coherence and tightness.

For the first step, we train GloVe vectors on a relatively large corpus consisting of data similar to the data we'll be analyzing. GloVe is an algorithm which uses co-occurrence statistics of the terms in a corpus with a weighted least-squares model in order to derive a vector for each term in a corpus such that similar terms are associated with vectors with high cosine similarity [15]. The authors of GloVe have different objectives (synonym detection and analogy completion) for their vectors and find that 300-dimensional vectors are optimal for their tasks. Such vectors are too large for our purposes. Since the ultimate goal of these vectors is to construct and cluster a set of vectors from C_t, the dimensionality of the vectors should be sufficiently small so as not to be hindered by the curse of dimensionality (i.e., the idea that as dimensionality grows, the distance between any two randomly chosen points on the unit sphere approaches $\sqrt{2}$).

In the second step, we derive a set of vectors to represent the content of the target corpus C_t. We describe the method we use here, although other methods are possible. For each document $d \in C_t$, we construct a "document vector" $v(d)$ by taking a weighted and normalized sum of the word vectors for words occurring in d. Explicitly, we define

$$\tilde{v}(d) = \sum_{w \in d} \mathrm{tf}_d(w)\, \mathrm{idf}_{C_0}(w)\, \mathbf{w}, \tag{6}$$

where $\mathbf{w}$ denotes the word vector associated with w obtained by the GloVe algorithm, $\mathrm{tf}_d(w)$ denotes the number of times the term w occurs in document d, and $\mathrm{idf}_{C_0}(w)$ denotes a smoothed version of inverse document frequency of w in C_0:

$$\mathrm{idf}_{C_0}(w) = \log\left(\frac{1 + |C_0|}{1 + |\{d \in C_0 \mid w \in d\}|}\right), \tag{7}$$

where $|\{d \in C_0 \mid w \in d\}|$ represents the number of documents in C_0 containing w.

We define the document vector for d as a normalized version of $\tilde{v}(d)$ defined by Eq. (6), that is:

$$v(d) = \frac{\tilde{v}(d)}{\|\tilde{v}(d)\|}. \tag{8}$$

This normalization reflects our belief that documents with similar content but differing lengths should be treated as similar. Finally, we use the set of document vectors $v(d)$ as our set of "corpus vectors."

In the third step, we cluster the corpus vectors. Because all of our vectors have unit length, standard Gaussian or Euclidean clusterings are not appropriate. Instead, we consider three variants of von Mises–Fisher (VMF) clustering, which are described at length in [2] and [3]. The VMF distribution is defined as the restriction to the unit sphere of a multivariate Gaussian distribution whose covariance matrix is a multiple of the identity. The probability density function of a VMF distribution with location μ (where $\|\mu\| = 1$) and concentration $\kappa \geq 0$ is given by:

$$p(x; \mu, \kappa) \propto \exp\left[\kappa \mu^{\intercal} x\right]. \tag{9}$$

The vector μ is analogous to the mean of a multivariate normal distribution, and the parameter $\kappa \geq 0$ is analogous to the inverse of the variance of a normal distribution. We consider three VMF mixture models:

1. Spherical k-means clustering. Spherical k-means clustering can be reinterpreted as a hard VMF mixture model where all mixture components are forced to have the same concentration [3].
2. Hard VMF mixture model. In this model, we fit to our data a mixture of VMF components with an underlying assumption that each data point can belong to only one mixture component.
3. Soft VMF mixture model. In this model, we fit to our data a mixture of VMF components with an assumption that each datum could have been drawn (with varying probability) from any mixture component.

In the fourth step, we construct scores which measure cluster coherence and tightness. The scores that we generate are dependent on which VMF mixture model we use. There are multiple such measures; we list here only the most promising:

– The concentration κ derived from reinterpreting spherical k-means clustering as a VMF mixture model.
– The median concentration parameter from both the hard and soft VMF mixture models. After fitting to our data a mixture model consisting of k mixture components, we collect the set of concentration scores. Empirical evidence suggests that the median of the concentration scores is higher on days when relatively few topics are receiving heightened interest. We also considered the first and third quartiles as potential scores.
– The lognormal location of the concentration parameters from the VMF mixture models. After constructing the set of concentration scores discussed above, we first discard any outlier concentration scores. Empirically, we have found that very high concentration scores result when we have a corpus with many highly similar documents. We next fit a lognormal distribution to the set of remaining concentration scores. A variable X has a lognormal distribution if $\ln(X) \sim \mathcal{N}(\mu, \sigma)$ (i.e., when $\ln(X)$ is normally distributed with mean μ and standard deviation σ). The values μ and σ are typically referred to as the location and scale of the lognormal distribution, respectively. We have found that the location parameter is typically higher on days when relatively few topics are receiving heightened interest, although this effect is more pronounced with a hard VMF mixture model than a soft VMF mixture model.

In all three methods detailed above, we rely on the techniques and formulae presented and explained in detail in [3, 20].

Remark In future work, we would like to incorporate various successful time-sensitive Bayesian topic models into our framework [1, 5]. Bayesian topic models are typically learned using one of two techniques—Gibbs sampling and variational inference. When trained with variational inference, Bayesian topic models provide *distributions* over parameter estimates. Just as we find the concentration scores of our von Mises–Fisher mixture models helpful in identifying anomalous time periods, so too could we utilize the covariance matrices of the posterior parameter distributions in our analysis. For example, we hypothesize that the variance $var(X)$ of each parameter X in the posterior distribution is inversely proportional to the trendiness exhibited by the set of documents in the corpus.

3.3 Weighted Probabilistic Fusion

In Sects. 3.1 and 3.2, we discussed numerous techniques for generating scores. In this section, we discuss a promising technique for fusing together various scores. Our technique is almost identical to that discussed in [17].

Empirical evidence suggests that our score generating techniques suffer from a lower-than-desired signal-to-noise ratio, and that the scores produced by any one technique are typically not normally distributed. As such, it would be inappropriate to use fusion techniques which return the weighted average or maximum of normalized scores as is done in other contexts. Instead, our fusion technique incorporates estimates of the various score distributions.

For each corpus C_t, we assume that we have generated m different scores $z_{t,1}, \ldots, z_{t,m}$ from one of the techniques discussed in Sects. 3.1 and 3.2. We assume that the values $\{z_{t,j}\}_t$ are sampled from some distribution Z_j with cumulative distribution function (cdf) F_j. Since the true cdf F_j is not known, we approximate F_j using either the empirical cdf $F_t^{(\text{emp})}$ or by using the cdf $F_t^{(\beta)}$ of a beta distribution fit to the scores $\{z_{t,j}\}_t$ (after scaling the $z_{t,j}$ to lie strictly between 0 and 1). Empirical evidence suggests that our fusion technique produces a greater number of significant events when using $F_t^{(\text{emp})}$ than when using $F_t^{(\beta)}$.

Our fusion technique involves three steps:

1. For all scores of type j, construct a cdf F_j as described above.
2. For each time period t, construct a fused score s_t via:

$$s_t = -\sum_{j=1}^{m} c_j \log\left(1 - F_j(z_{t,j})\right), \tag{10}$$

where $c_j > 0$ denotes the relative weight we wish to give the jth score generating technique.

3. Fit a gamma distribution with cdf G to the set of fused scores $\{s_1, \ldots, s_n\}$. For any given time period t, the value $G(s_t)$ now quantifies the significance of the events occurring during t.

Our model assumes stationarity; that is, each cdf F_j is assumed to be time invariant. If our data spans a sufficiently large period of time, this assumption may be inappropriate. In such circumstances, we modify step (1) above and fit a separate cdf $F_{t,j}$ for each score j and time period t from the scores $\left\{z_{\tau,j}\right\}_\tau$, where τ ranges over a set of time periods which are temporally proximal to the target time period t. In step (2), we then calculate s_t using the cdfs $\left\{F_{t,j}\right\}_j$. Step (3) remains unchanged. We call the fusion technique described in this paragraph "windowed fusion" in contrast to the original "global fusion" technique presented in the enumerated list above.

We now give a rough justification of our empirically successful fusion method, recognizing that the assumptions made in our justification may be invalid in a real-world scenario. If we assume that the set of scores $\left\{z_{t,j}\right\}_{t,j}$ have been independently sampled (where $z_{t,j}$ has been sampled from a distribution with cdf F_j), then the values $\left\{-\log\left(1 - F_j(z_{t,j})\right)\right\}_{t,j}$ are iid samples from an exponential distribution. If we additionally assume that $c_j = 1$ for all j, then the values s_t are iid samples from a gamma distribution (because the sum of independent exponential random variables is a Gamma random variable).

We note that, in general, each F_j only approximates the true cdf of the corresponding score distribution, and, for any fixed time period t, the scores $\left\{z_{t,j}\right\}_j$ are far from independent. In fact, we rely on the assumption that during an anomalous time period t, all $z_{t,j}$ will be abnormally high. Furthermore, we may want to choose our score weights c_j to be nonuniform. In our experiments, we often choose c_j so that the scores generated from term distribution analysis (Sect. 3.1) have combined weight equal to that of the scores generated by analyzing cluster coherence (Sect. 3.2).

4 Experiments

Our overall motivating goal—finding content-based anomalies in temporal segments of a corpus of social media posts—is somewhat vague and underspecified. We have thus chosen to focus our analysis on the somewhat more tractable goal of finding time periods exhibiting unusually high trendiness. Yet even with this specification, we suffer not only from a lack of a clear and unambiguous definition of "trendiness" (although we have chosen to use an information-theoretic definition of "anomaly" and cluster coherence as proxies), but also from the absence of data with incontrovertible ground truth with labeled anomalous time periods. Nevertheless, we present the results of applying our methods on multiple diverse Twitter datasets to demonstrate the capabilities of the proposed algorithm.

4.1 Data

We first apply our algorithm to relatively small subsamples of the Twitter Streaming API, a free public stream consisting of social media posts containing at most 140 characters. In total, four datasets are considered. The first, referred to as *TwitterParisEnglish*, consists of 50,000 tweets per day sampled uniformly at random from all English tweets from the Twitter Streaming API from October 11, 2015 to November 29, 2015. The second dataset, *TwitterParisFrench*, consists of 53,000 tweets per day sampled uniformly at random from all French tweets from the Twitter Streaming API from October 16, 2015 to November 29, 2015. Note that the sampling period for both these datasets includes both November 13, 2015, the date of major terrorist attacks in Paris, France, and November 26, 2015, the date of the US holiday Thanksgiving.

We next apply our algorithm to datasets consisting of all tweets emitted by specified users during a specified timeframe constructed using the Twitter Search API. In particular, we construct a dataset *TwitterUSUniversities* by collecting all 4.2 million tweets emitted from official Twitter accounts of 2300 US universities from May 2014 to December 2016. We further construct a dataset *TwitterOlympics* by collecting all 1.1 million tweets emitted from the accounts of 1200 Olympians and Olympics professionals (e.g., coaches, and sports journalists) from October 2014 to December 2016.

For the analysis of all our Twitter datasets except *TwitterParisFrench*, we use 25-dimensional GloVe vectors trained on roughly 50 million English tweets sampled from the Twitter Streaming API from March, 2015 to July, 2015. For *TwitterParisFrench*, we use 25-dimensional GloVe vectors trained on roughly 5 million French tweets sampled from the Twitter Streaming API from January, 2015 to August, 2015. Note that the GloVe vectors we use are trained on tweets temporally separated from the *TwitterParisEnglish* and *TwitterParisFrench* datasets by a period of at least two months. We also feel that *TwitterUSUniversities* and *TwitterOlympics* are largely independent from the data used to train the GloVe vectors.

It has been shown that changes in the usage of common words such as “me” and “you” can be indicative of public sentiment. For example, previous research [14, 22] suggests that usage of plural pronouns among members of a society increases after a society-scale attack due to the unification of members in the attacked society against a common threat. Therefore, in all cases, common stopwords were included in our analysis.

4.2 Results

We first run a PAKL analysis (cf. Sect. 3.1) for our *TwitterParisEnglish* dataset using the “document frequency” option. We segment our corpus by day, and for

Table 1 Top words for select days and their associated PAKL scores from *TwitterParisEnglish*

Oct. 26, 2015		Nov. 4, 2015	
forevermore	0.0286	#aldub16thweeksary	0.0203
#pushawardslizquens	0.0154	i	0.0110
#aldubpredictions	0.0149	#showtimehousemates	0.0103
the	0.0149	that	0.0085
#aldubnewbeginnings	0.0129	it	0.0083
everydayiloveyou	0.0142	#otwolmanilainlove	0.0082
#everydayilov…	0.0105	to	0.0078
#otwolhappytimes	0.0104	#cmaawards	0.0076
i	0.0096	#aldubnewcharacter	0.0076
you	0.0092	a	0.0075
Nov. 13, 2015		Nov. 26, 2015	
paris	0.1448	thanksgiving	0.1743
in	0.0682	thankful	0.1159
#prayforparis	0.0582	happy	0.0692
the	0.0572	#mtvstars	0.0602
#madeintheam	0.0485	for	0.0402
#aldubhappybdaylola	0.0392	britney	0.0343
is	0.0381	spears	0.0342
#paris	0.0341	rey	0.0322
and	0.0312	lana	0.0321
prayers	0.0307	del	0.0315

the analysis of day t, we consider only terms which occur at least 5 times in C_t and 20 times in the entire corpus. The terms with the highest PAKL scores for select days can be seen in Table 1. We include terms from both uneventful days (Oct. 26, 2015 and Nov. 4, 2015) and anomalous days (Nov. 13, 2015 and Nov. 26, 2015). For the anomalous days, we can successfully find terms of interest. Note also that the top PAKL scores for anomalous days tend to be higher than those for normal days.

We also wish to mention that on November 26, 2015, roughly 2% of our tweets mention “#mtvstars,” “Britney Spears,” and “Lana Del Rey.” A post hoc analysis has revealed that the vast majority (over 98%) of these tweets were posted by accounts that are now suspended for violating the Twitter Rules. Even so, other terms associated with Thanksgiving, including “family,” “turkey,” and “#imthankfulfor,” are included in the 20 highest scoring terms for November 26, 2015.

We also score each document $d \in C_t$ using the term PAKL scores for C_t via:

$$\text{score}(d) = \frac{\ln(|d|)}{|d|} \sum_{w \in d} \text{PAKL}(w). \tag{11}$$

We report the top two documents for select days in Table 2. For anomalous days, these documents successfully capture the nature of the day’s anomaly.

Table 2 Top documents for select days from *TwitterParisEnglish*

Oct. 26, 2015	Nov. 4, 2015
EVERYDAYILOVEYOU Forevermore in the night #PushAwardsLizQuens	I'm chillin I'm good I'm straight
I LOOVE EVERYDAYILOVEYOU Forevermore #PushAwardsLizQuens	I don't know, that that's a thing that I know
Nov. 13, 2015	**Nov. 26, 2015**
Sending prayers to the people in Paris #PrayForParis	thankful for everything <emoji> Happy Thanksgiving
My thoughts and prayers go out the victims in the shootings in Paris #Prayers4Paris	<emoji> Happy Thanksgiving <emoji>

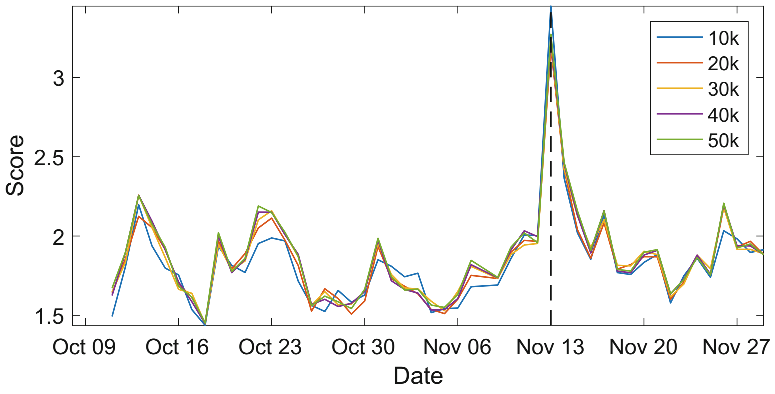


Fig. 1 PAKL scores per day for corpora of varying size taken from *TwitterParisEnglish*

In order to test our methods' robustness to corpora of different sizes, we create subcorpora of *TwitterParisEnglish* containing 10,000, 20,000, 30,000, and 40,000 tweets per day. We plot the sum of all positive PAKL scores for each day in Fig. 1. We find that varying the number of tweets considered causes surprisingly little variation in the score. Similarly, we plot the concentration score for spherical k-means clustering (with $k = 50$) in Fig. 2. Although the clustering scores are less robust to the number of tweets considered each day than the PAKL scores, they still maintain a level of robustness sufficient to identify anomalous time periods with high confidence.

Figure 3 shows the first, second, and third quartiles of the concentration scores for a hard VMF mixture model with 50 mixture components for the *TwitterParisEnglish* dataset. Similar graphs, not shown here, were produced for the other datasets mentioned above. We normalize the scores from each quartile for a fair comparison of score quality.

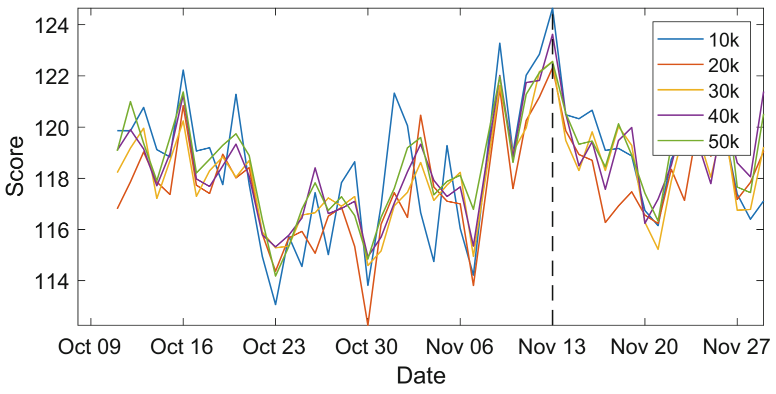


Fig. 2 Cluster scores per day for corpora of varying size taken from *TwitterParisEnglish*

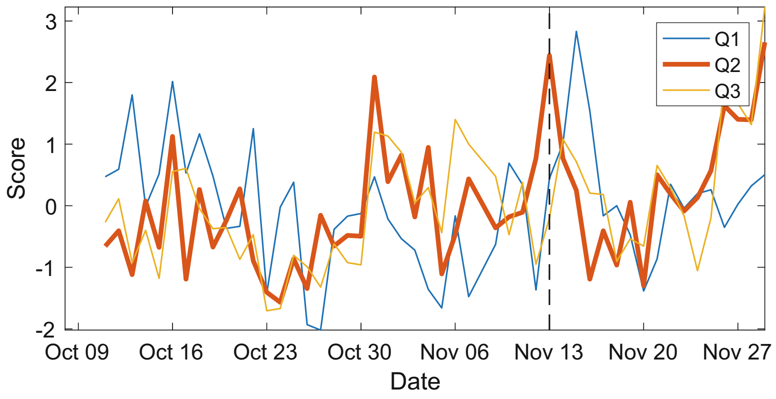


Fig. 3 VMF mixture cluster scores per day for *TwitterParisEnglish*

We have experimented with varying the number of clusters k between 10 and 100. For k in this range, the effects of changing k are noticeable but relatively insignificant. In general, as k decreases, clustering scores become both more resilient to changing dataset size and less noisy (the randomness inherent in many clustering algorithms creates a lack of uniformity in clustering scores across different clusterings of the same data). Unfortunately, the quality of the clustering scores also tends to decrease with decreasing k.

We also present graphs produced by fusing PAKL scores with clustering scores. Unless otherwise noted, fusion for these datasets is done via $F^{(\beta)}$, and the cdfs used during fusion are calculated from the entire dataset, rather than from windows around the target time periods.

For the *TwitterParisEnglish* and *TwitterParisFrench* datasets, we construct four PAKL scores by summing, for each day, all PAKL scores, all positive PAKL scores, the highest 200 PAKL scores, and the highest 50 PAKL scores. We also construct

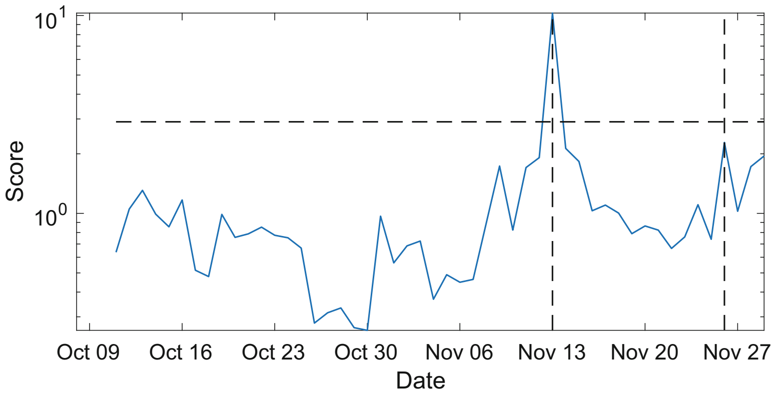


Fig. 4 Fused scores per day for *TwitterParisEnglish*

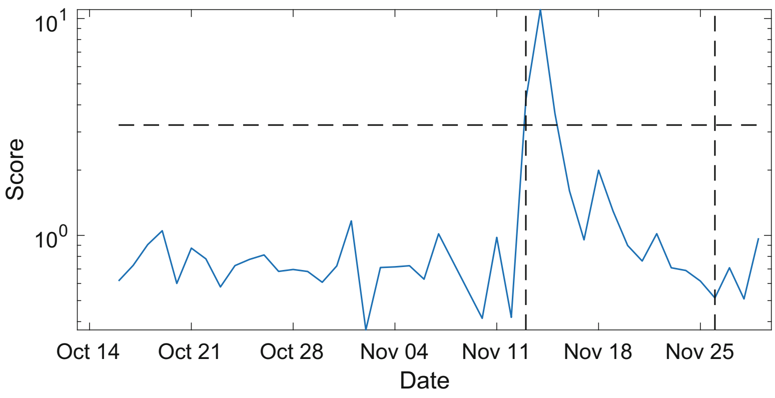


Fig. 5 Fused scores per day for *TwitterParisFrench*

fifteen cluster scores: we run each clustering algorithm (spherical k-means, hard VMF, and soft VMF) three times with $k = 50$ clusters. From the spherical k-means clusterings, we record the concentration. From the VMF mixture models, we collect the lognormal location parameter and the median concentration. We weight the scores so that the PAKL and clustering scores each account for 50% of the total fused score. The fused scores for *TwitterParisEnglish* (resp., *TwitterParisFrench*) are shown in Fig. 4 (resp., Fig. 5). Dashed vertical lines denote the date of the Paris attacks and Thanksgiving, and a dashed horizontal line indicates the 10% significance level.

For the *TwitterOlympics* and *TwitterUSUniversities* datasets, fusion is performed similarly with the following changes to account for the fact that these corpora are smaller in general than *TwitterParisEnglish* and *TwitterParisFrench*. First, we segment these corpora by week rather than by day. We also construct four PAKL scores, but construct scores by summing the highest 100 and 20 PAKL scores instead

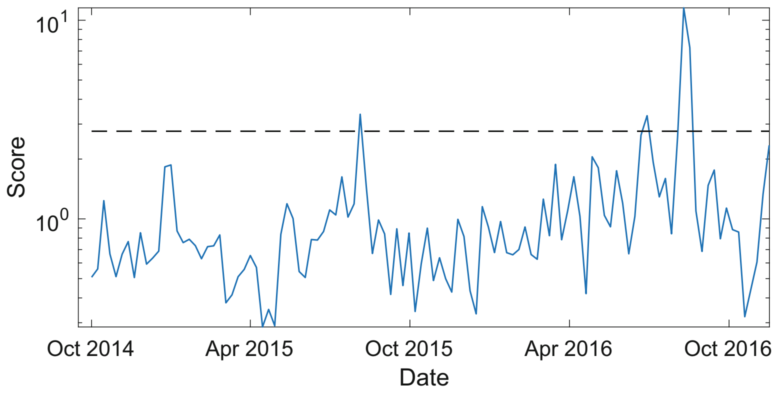


Fig. 6 Fused scores for *TwitterOlympics* using $F^{(\beta)}$

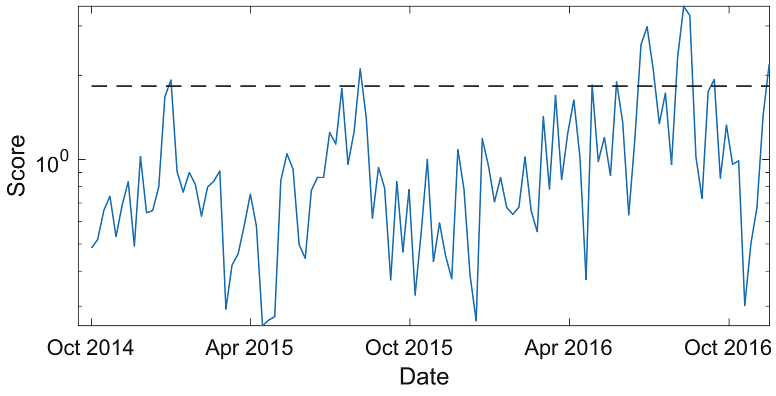


Fig. 7 Fused scores for *TwitterOlympics* using $F^{(\text{emp})}$

of the highest 200 and 50 scores as above. Finally, we run our clustering score generators with $k = 25$ instead of $k = 50$. We again use dashed horizontal lines to indicate the 10% significance level.

For *TwitterOlympics*, we produce fused scores using both $F^{(\beta)}$ (Fig. 6) and $F^{(\text{emp})}$ (Fig. 7). Although these graphs have very similar shapes, fusion using $F^{(\beta)}$ tends to produce fewer significant events than fusion using $F^{(\text{emp})}$. The three periods in Fig. 6 with significant scores correspond to the various athletic events in August 2015, the 2016 Olympic trials, and the 2016 Summer Olympics.

For *TwitterUSUniversities*, we present a graph of the fused scores based on the entire corpus (Fig. 8). We note that the Twitter feeds of many US universities changed drastically between May 2014 and November 2016. Although our algorithms are robust to changing corpus size and sampling rates, they are not robust to underlying changes in *behavior*. For example, the rate of tweet production nearly triples throughout our period of collection. Although this first appears to be a change

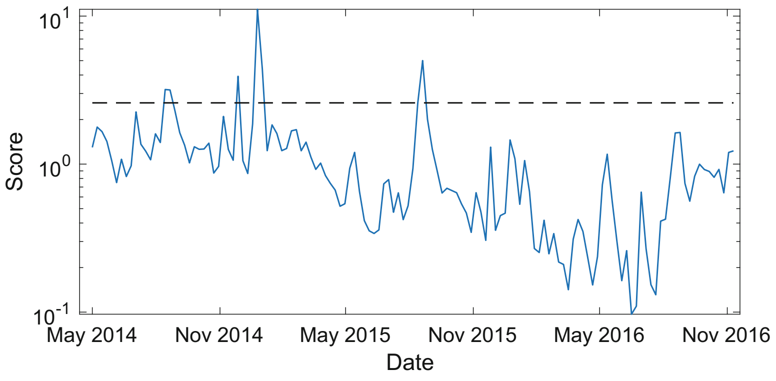


Fig. 8 Fused scores for *TwitterUSUniversities* during 2014–2016 using global fusion

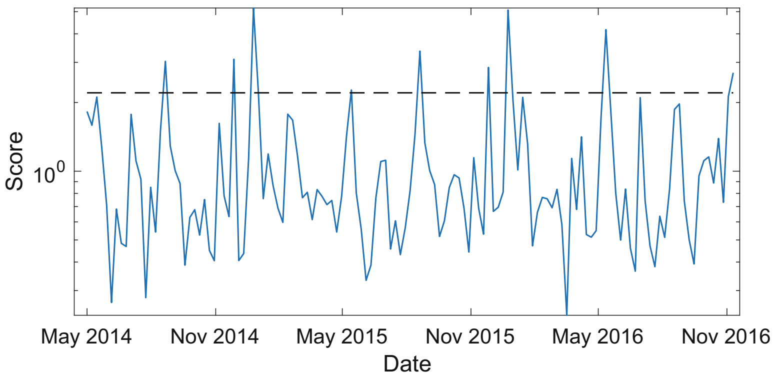


Fig. 9 Fused scores for *TwitterUSUniversities* during 2014–2016 using windowed fusion

in corpus size, we see upon further inspection that it is a change in behavior, and thus, a violation of the assumption of stationarity—later in our collection period, universities are more likely to tweet about less pressing matters, so significant events receive less attention in general. Consequently, no time periods register as significant in the latter temporal half of this corpus when using global fusion.

However, a much clearer pattern emerges when using windowed fusion to analyze *TwitterUSUniversities* (Fig. 9). For this analysis, we fit cdfs $F_{t,j}$ for the jth score generating technique and time period t from the scores generated by the jth score generating technique for the 15 time periods before t and the 15 time periods after t. With this modification, we see peaks for both the 2014–2015 school year and the 2015–2016 school year corresponding to the beginning of the school year, Thanksgiving break, Winter break, and the end of the school year.

We note that we have found it beneficial to fuse the clustering scores with the PAKL scores, rather than relying on either alone. For example, the first peak in Fig. 6 corresponding to the August 2015 athletic events can be attributed more to clustering scores than PAKL scores. During this event, PAKL scores barely rise above baseline; since each sport has its own world championship, the difference in term distribution from baseline is no more than expected. However, cluster coherence is particularly high during this timeframe due to the large percentage of tweets related to competition.

5 Conclusion

We have introduced two techniques which merge anomaly detection with topic detection and tracking. Our first technique relies on an information-theoretic examination of the term distributions of corpora collected over time. Our second approach produces a set of values which serve as measures for the homogeneity of the contents of the corpus. For sufficiently large corpora, both techniques are agnostic to the size of the corpus. We then explain how the scores produced from our techniques can be combined to form a single summary score. We demonstrate our algorithms on various Twitter datasets and conclude that our techniques are successful in identifying portions of a corpus with unusual and interestingly high trendiness.

References

1. AlSumait, L., Barbará, D., Domeniconi, C.: On-line LDA: adaptive topic models for mining text streams with applications to topic detection and tracking. In: Proceedings of the 2008 Eighth IEEE International Conference on Data Mining, pp. 3–12. IEEE, Piscataway (2008)
2. Banerjee, A., Dhillon, I., Ghosh, J., Sra, S.: Generative model-based clustering of directional data. In: Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 19–28. ACM, New York (2003)
3. Banerjee, A., Dhillon, I.S., Ghosh, J., Sra, S.: Clustering on the unit hypersphere using von Mises–Fisher distributions. J. Mach. Learn. Res. **6**, 1345–1382 (2005)
4. Blei, D.M., Ng, A.Y., Jordan, M.I.: Latent Dirichlet allocation. J. Mach. Learn. Res. **3**, 993–1022 (2003)
5. Blei, D.M., Lafferty, J.D.: Dynamic topic models. In: Proceedings of the 23rd International Conference on Machine learning, pp. 113–120. ACM, New York (2006)
6. Cheng, Z., Caverlee, J., Lee, K.: You are where you tweet: a content-based approach to geo-locating Twitter users. In: Proceedings of the 19th ACM International Conference on Information and Knowledge Management, pp. 759–768. ACM, New York (2010)
7. Golbeck, J., Robles, C., Turner, K.: Predicting personality with social media. In: CHI'11 Extended Abstracts on Human Factors in Computing Systems, pp. 253–262. ACM, New York (2011)
8. Griffiths, T.L., Steyvers, M.: Finding scientific topics. Proc. Natl. Acad. Sci. **101**(Suppl. 1), 5228–5235 (2004)
9. He, D., Parker, D.S.: Topic dynamics: an alternative model of bursts in streams of topics. In: Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 443–452. ACM, New York (2010)

10. Kleinberg, J.: Bursty and hierarchical structure in streams. In: Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 91–101. ACM, New York (2002)
11. Lee, K., Palsetia, D., Narayanan, R., Patwary, M.M.A., Agrawal, A., Choudhary, A.: Twitter trending topic classification. In: Proceedings of the 2011 IEEE 11th International Conference on Data Mining Workshops, pp. 251–258. IEEE, Piscataway (2011)
12. Leskovec, J., Backstrom, L., Kleinberg, J.: Meme-tracking and the dynamics of the news cycle. In: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 497–506. ACM, New York (2009)
13. Morinaga, S., Yamanishi, K.: Tracking dynamics of topic trends using a finite mixture model. In: Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 811–816. ACM, New York (2004)
14. Pennebaker, J.W., Mehl, M.R., Niederhoffer, K.G.: Psychological aspects of natural language use: our words, our selves. Annu. Rev. Psychol. **54**(1), 547–577 (2003)
15. Pennington, J., Socher, R., Manning, C.D.: GloVe: global vectors for word representation. In: Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP), vol. 14, pp. 1532–1543 (2014)
16. Piantadosi, S.T.: Zipf's word frequency law in natural language: a critical review and future directions. Psychon. Bull. Rev. **21**(5), 1112–1130 (2014)
17. Simonson, K.: Probabilistic fusion of ATR results. Tech. Rep. SAND98–1699. Sandia National Laboratories (SNL-NM), Albuquerque, NM (1998)
18. Skryzalin, J., Field, R., Fisher, A., Bauer, T.: Temporal anomaly detection in social media. In: Proceedings of the IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining, pp. 505–508. ACM, New York (2017)
19. Spinosa, E.J., de Leon F de Carvalho, A.P., Gama, J.: OLINDDA: a cluster-based approach for detecting novelty and concept drift in data streams. In: Proceedings of the 2007 ACM Symposium on Applied Computing, pp. 448–452. ACM, New York (2007)
20. Sra, S.: A short note on parameter approximation for von Mises–Fisher distributions: and a fast implementation of $i_s(x)$. Comput. Stat. **27**(1), 177–190 (2012)
21. Swan, R., Allan, J.: Extracting significant time varying features from text. In: Proceedings of the 8th International Conference on Information Knowledge Management, pp. 38–45. ACM, New York (1999)
22. Tausczik, Y.R., Pennebaker, J.W.: The psychological meaning of words: LIWC and computerized text analysis methods. J. Lang. Soc. Psychol. **29**(1), 24–54 (2010)
23. Tumasjan, A., Sprenger, T.O., Sandner, P.G., Welpe, I.M.: Predicting elections with Twitter: what 140 characters reveal about political sentiment. ICWSM **10**(1), 178–185 (2010)
24. Wang, X., McCallum, A.: Topics over time: a non-Markov continuous-time model of topical trends. In: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 424–433. ACM, New York (2006)
25. Wang, C., Blei, D., Heckerman, D.: Continuous time dynamic topic models. In: Uncertainty in Artificial Intelligence (UAI). pp. 579–586 (2008)
26. Yang, J., Leskovec, J.: Patterns of temporal variation in online media. In: Proceedings of the Fourth ACM International Conference on Web Search and Data Mining, pp. 177–186. ACM, New York (2011)
27. Zhang, X., Shasha, D.: Better burst detection. In: Proceedings of the 22nd International Conference on Data Engineering, p. 146. IEEE, Piscataway (2006)
28. Zhu, Y., Shasha, D.: Efficient elastic burst detection in data streams. In: Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 336–345. ACM, New York (2003)

Index

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