

OCTracker: A Density-Based Framework for Tracking the Evolution of Overlapping Communities in OSNs

Sajid Yousuf Bhat

Department of Computer Science
Jamia Millia Islamia, New Delhi, India
Email: s.yousuf.bhat@gmail.com

Muhammad Abulaish, *SMIEEE*

Center of Excellence in Information Assurance
King Saud University, Riyadh, Saudi Arabia
Email: abulaish@ieee.org

Abstract—In this paper, we propose a unified framework **OCTracker** for tracking overlapping community evolution in online social networks. **OCTracker** adapts a preliminary community structure towards dynamic changes in social networks using a novel density-based approach for detecting overlapping community structures and automatically detects evolutionary events like *birth*, *growth*, *contraction*, *merge*, *split*, and *death* of communities with time. Unlike other density-based community detection methods, the proposed method does not require the neighborhood threshold parameter to be set by the users, rather it automatically determines the same for each node locally.

Index Terms—Data mining; Social network analysis; Community detection; Community evolution; Density-based clustering.

I. INTRODUCTION

Due to increasing popularity of Online Social Networks (OSNs) and its wide application areas, community mining research has received a lot of attention in recent past and the field is still rapidly evolving. Numerous methods based on spectral clustering, partitional clustering, modularity optimization and latent space clustering have been developed for community detection in social networks. One of the important properties of real-world social networks is that they tend to change dynamically and this property has until recently been largely ignored in terms of community detection. In case of dynamic social networks, most of the studies either analyze a single snapshot of the network or an aggregation of all interactions over a possibly large time-window. But, such approaches may miss important tendencies of dynamic networks and in fact the possible causes of this dynamic behavior may be among the most important properties to observe [1]. The main phenomena occurring in the lifetime of communities that have become desirable to be investigated are *birth*, *expansion*, *shrinkage*, *merge*, *split*, and *death*. Although, recent literature includes some approaches for analyzing communities and their temporal evolution in dynamic networks, a common weakness in these studies is that communities and their evolutions are studied separately. As pointed out in [2], a more appropriate approach would be to analyze communities and their evolution in a unified framework, where community structure provides evidence about community evolution.

In this paper, we propose the design of a unified framework, **OCTracker**, which exploits a density-based approach to track the evolution of overlapping community structures in OSNs. The proposed approach is in line with **SCAN**, **DENGRAPH** and other recent density-based community detection methods that are based on **DBSCAN**. These methods find dense communities and also detect outliers in networks. Besides these properties, the proposed method has the following additional features.

- **OCTracker** detects dynamic overlapping community structures by automatically highlighting evolutionary events like *birth*, *death*, *split* and *shrinkage/expansion* with time using a density-based approach.
- **OCTracker** does not need the neighborhood threshold ε (mostly difficult to determine for density-based community detection methods) to be specified by the users manually.
- **OCTracker** is computationally faster and naturally scalable to large social networks.

II. RELATED WORK

Falkowski et al. [3] extended the **DBSCAN** algorithm [4] to weighted interaction graph structures of OSNs. Some important features of density-based community detection methods include efficiency, outlier detection, and natural scalability to large networks. However, the main drawback such methods is the requirement of a global neighborhood threshold ε and the minimum cluster size μ that need to be specified by the users. The methods are particularly sensitive to the ε parameter, which is difficult to determine. The most popular method for identifying overlapping communities is the Clique Percolation Method (CPM) proposed by Palla et al. [5], which is based on the concept of k -clique. In order to find communities in dynamic social networks and to track their evolutions, various methods have been proposed recently. A typical dynamic community detection problem is formulated in [1]. In this work, along a discrete timescale and at each time-step, social interactions of certain individuals of a network are observed and several subgraphs are formed. Based on these subgraphs, the true underlying communities and their developments over

TABLE I: Notations and their descriptions

Notation	Description
V	The set of nodes in the social network
$I_{\vec{p}}$	Total number of out-going interactions of a node p
$I_{\vec{p}q}$	Number of interactions from node p to node q
$I_{\vec{p}q}$	Reciprocated interactions of p and q : $\min(I_{\vec{p}q}, I_{\vec{q}p})$
$I_{\vec{p}}$	Number of reciprocated interactions of a node p : $\sum_{q \in V_p} \min(I_{\vec{p}q}, I_{\vec{q}p})$
V_p	Set of nodes in the networks with whom node p interacts
V_{pq}	Set of nodes with whom both nodes p and q interact: $V_p \cap V_q$

time are identified, so that most of the observed interactions can be explained by the inferred community structure. Similar approaches have been followed in [6], [7]. However, as pointed out in [2], a common weakness of these approaches is that communities and their evolution are studied separately. It would be more appropriate to analyze communities and their evolution in a unified framework where community structure provides evidence about the community evolutions. Along this direction, Falkowski et al. [8] proposed a framework for studying community dynamics where a preliminary community structure adapts to dynamic changes in a social network. Our approach is similar to [8], but unlike it, our concern is on tracking the evolution of overlapping communities and we do not need an ageing function to remove old interactions from the network. Moreover, our proposed framework is applicable to directed/un-directed and weighted/un-weighted networks, whereas [8] applies only to un-directed and weighted networks.

III. PROPOSED METHOD

As pointed out in section II, the main drawback of traditional density-based community detection methods is that they require a global neighborhood threshold ε and a minimum cluster size μ to be specified by the users. The proposed method does not require the global neighborhood threshold parameter ε to be set manually at the beginning of the process. Instead, it uses a local representation of the neighborhood threshold which is automatically calculated for each node locally using a much simpler approach from the underlying social network. Moreover, a local version of μ is also computed for each node automatically using a global percentage parameter η . The proposed method thus requires only a single tunable parameter η to be set by the users. Before discussing density-based overlapping community detection method, first we present some preliminary definitions. The notations used in these definitions are described in table I.

Definition 1 (Response). *For any two nodes $p, q \in V$ in an interaction graph $G_I = \langle V, E_w \rangle$ of a social network, the response of the node q and the commonly interacted nodes of p and q , to the interactions of node p , is represented as $response(p, q)$ and defined as the average of the per-user reciprocated interactions of q and the common nodes of p and q with p using equation 1.*

$$response(p, q) = \begin{cases} \left(\frac{\sum_{s \in V_{pq}} (I_{\vec{p}s}) + I_{\vec{p}q}}{|V_{pq}| + 1} \right) & \text{if } I_{\vec{p}q} > 0 \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

Definition 2 (Distance). *For any two interacting nodes $p, q \in V$, the distance between them is represented as $dist(p, q)$ and defined as the minimum of the reciprocals of their mutual directed responses, normalized by their respective total count of outgoing interactions in the interaction graph, as shown in equation 2.*

Definition 3 (Local-Neighborhood Threshold (ε_p)). *For a node $p \in V$, the local-neighborhood threshold is represented as ε_p and defined using equation 3 as the average per-receiver reciprocated interaction-score of p with all its neighbors (i.e., friends and non-friends with which it interacts).*

$$\varepsilon_p = \begin{cases} \left(\frac{I_{\vec{p}}}{|V_p|} \right)^{-1} & \text{if } I_{\vec{p}} > 0 \\ 0 & \text{otherwise} \end{cases} \quad (3)$$

Definition 4 (Local ε_p -neighborhood (N_{localp})). *The local ε_p -neighborhood of a node $p \in V$ is represented by N_{localp} and defined as the set of p 's interacting nodes whose distance with p is less than or equal to ε_p (see equation 4).*

$$N_{localp} = \{q : q \in V_p \wedge dist(p, q) \leq \varepsilon_p\} \quad (4)$$

For proposed method, we define a local version of minimum-number-of-points for a node p , represented by μ_p , which is also computed automatically from the underlying network. However, we need to specify a *global percentage threshold* (η) to compute μ_p . For a node $p \in V$, μ_p is taken as η percent of its interacted nodes in the network. It should be noted that the global percentage constant (η) forms the only parameter for the proposed method to be set by the users. Moreover, besides determining the local minimum-number-of-points threshold (μ_p) for a node p , the value of η is also used to specify a distance constraint, which is specified as follows: The distance between two interacting nodes p and q can be measured by equation 2 only if the number of commonly interacted nodes of p and q is greater than η percent of the minimum of their individually interacted nodes minus one; otherwise, it is taken as 1.

A. COMMUNITY DETECTION PROCESS

For a given global percentage threshold (η), the community detection process iteratively finds a density-connected community by randomly selecting an un-visited node, say p , to grow a community using density-reachable relationship of p with other nodes. It checks whether p is a core node and if p qualifies the test, it finds all density-reachable nodes of p to identify its community. To do so, it first computes the local ε_p threshold for p using equation 3. If the ε_p threshold for p is greater than zero, then it uses the distance function of

$$dist(p, q) = \begin{cases} \min\left(\frac{response(p,q)^{-1}}{I_{\vec{p}}}, \frac{response(q,p)^{-1}}{I_{\vec{q}}}\right) & \text{if } response(p, q) > 0 \wedge response(q, p) > 0 \\ 1 & \text{otherwise} \end{cases} \quad (2)$$

equation 2 and distance constraint to determine the local ε_p -neighborhood of p , i.e., $N_{local}p$. If node p qualifies as a core node, the following steps are performed to identify a density-connected community of p .

- 1) All un-visited mutual-core nodes of p in $N_{local}p$ are appended with the current community label. They are marked as visited and pushed to a stack to identify the density-reachable nodes of p . This step is later repeated for each node in the stack for the current community in order to find the connected sequences of mutual-core nodes starting from p . These connected sequences of mutual-core nodes form the *Mutual-core Connected Maximal Sub-graph* (MCMS) of a community. All nodes in the MCMS of a community are called the primary-core nodes of that community. However, if a core-node p does not show mutual-core relation with any other core-node, then only the node p along with its $N_{local}p$ forms a community with p being its only primary core-node.
- 2) If a core-node q in $N_{local}p$ is not a mutual-core of p , it is appended with the current community label; however, it is not pushed into the stack to grow the current community and its visited/un-visited status is kept un-altered.
- 3) Non-core nodes in $N_{local}p$ are marked as visited and they are appended with the current community label. Such nodes form boundary nodes for the community of p and are thus not pushed into the stack as they cannot be used to grow a community.

The steps through 1 – 3 are repeated for each un-visited node to find the overlapping community structure in the social network. At the end of the process, un-labeled nodes (if any) represent outlier nodes, i.e., they do not belong to any community as they do not show an interaction behavior that is similar to any node or enough number of nodes in the network.

IV. TRACKING EVOLUTIONARY EVENTS

It should be noted that unlike [8], we do not need an ageing function to remove old interactions and we also argue that it is difficult to decide upon a selection criteria to do so. As our approach involves local average interactions of nodes for the clustering process, addition of new interactions results in new averages for the involved nodes and directly effects their neighborhoods and roles for clustering. A social network and its resulting community structure can evolve due to various events triggered by the social network individuals. These events may include (i) addition of new weighted interaction links and/or nodes, (ii) increase in the interaction weights of existing links, and (iii) removal of existing nodes.

In order to track the evolution of communities in dynamic social networks like OSNs, the proposed framework OCTracker first detects a preliminary community structure

from an initial state of the network using the method discussed in section III-A. Then for each node involved in a change in the network, various transitions can occur. They can be handled by either considering a live stream of changes as the network evolves (an online evolutionary adaption of the community structure), or the set of changes observed in a specific time-window (an offline evolutionary adaption of the community structure). In either case, the new edges and/or nodes are added to the network or nodes are removed from the network, and each node involved in a change is marked as un-visited. Thereafter, each remaining un-visited node is re-checked for a core-node property by re-calculating its local $\varepsilon_{(p)}$ -neighborhood. Various events or transitions used by OCTracker to model the evolution of communities are presented in the following sub-sections.

A. A Non-Core Node Becomes a Core

In this case, either an existing non-core node or a newly added node in the network becomes a core node. In order to track a possible evolutionary event, OCTracker checks the following conditions. For the new core node p , if there exist core nodes in the local $\varepsilon_{(p)}$ -neighborhood with which the node p has mutual-core relations and which already belong to different communities, then p causes the primary communities of these core nodes to *merge* into a single community. Consequently, in this case, p causes the MCMSs of different communities to join and form a single MCMS for the new merged community. The merged community also forms the primary community of the new core node p and nodes in its local neighborhood are also added to the merged community. If the new core node p has mutual-core relations with nodes that have the same primary community C , then p also forms a primary core of community C by appending this community label to itself and to its local neighborhood. This simply results in the *expansion* of community C . Finally, if the new core node p has no mutual-core relations, then p forms a new community and appends the new community label to its local neighborhood and itself. This causes the *birth* of a new community with p being its only primary core.

B. A Core Node Becomes a Non-Core

In this case, an existing core node no longer remains a core node due to some change in the network. This triggers either a *split* or a *shrink* event in the evolution of a community as follows. Let p be a primary core node of a community C at a previous stage, and p cease to exist as a core node due to a new dynamic change in the network. Let S be the set of primary cores of the community C which had mutual-core relations with p before the change in the network. We mark the nodes in S as un-visited. For any core node $q \in S$, let T be a simple BFS traversal of nodes starting from q , visiting nodes

in the local neighborhoods of the core nodes and branching at mutual-core relations wherein each newly visited node is labeled as visited. If T includes all the core nodes in S , then p is simply removed from being a primary core of community C . Moreover, if p and/or any other node that belonged to the earlier local neighborhood of p are not in the traversal T , then they are removed with the community label of C , causing C to *shrink*. However, If T does not include all the core nodes in S , then T forms a new community, i.e., the original community C *split* as p with loosed core-node property causes a cut in the MCMS of C . The community label C of the nodes in T (which now represents a split part of community C) are replaced with a new community label. The traversals are repeated for each remaining un-visited core nodes in S until no further split of community C is possible, i.e., no node in S remains un-visited after a traversal. In the last traversal, if a node s is visited which does not have the community label of C (i.e., it was removed as s belonged to a previous traversal that split the community C), then the community label of C is re-appended to it resulting in an overlapping node. At the end, the node p and/or any node that belonged to its previous local neighborhood may be labeled with community label C , but do not belong to the last traversal. In this case, the community label C for these nodes is removed, causing community C to further *shrink*.

C. A Core Node Gains or Looses Nodes but Remains a Core

In this case, the addition or removal of nodes are handled as follows. If the local ε_p -neighborhood of a core node p gains a set of nodes S that do not have *mutual-core* relation with p , then the *primary-community* label of p is simply appended to each node $q \in S$. However, if the added nodes have *mutual-core* relation with p , then they are handled in the same way as the *mutual-cores* of a newly formed core node are handled (section IV-A). This can either cause the *expansion* of a community or *merge* of multiple communities. It is obvious that if all the *mutual-cores* of p in its neighborhood including p have the same *primary-community*, then only the neighborhood of p is updated resulting in *expansion* of a community.

Consider the case when the local ε_p -neighborhood of a core node p with a *primary-community* C looses a set of nodes L that were earlier in its ε_p -neighborhood. If the nodes in L do not have *mutual-core* relation with p , and they are not direct density-reachable from any other *primary-core* of the community C , then the community label of C is removed from the lost nodes resulting in the *shrinkage* of community C . However, if a core node p looses a set of nodes S that had *mutual-core* relation with it, then such nodes are handled in the same way when the *mutual-core* of a core node no longer remains a core node (section IV-B). But, in this case the core node p in question is not excluded from the set of nodes S . This could possibly lead to either *split* or no change to the community C .

V. EXPERIMENTAL RESULTS

In this section, we present the results of OCTracker on a benchmark dataset. We compare the results obtained through proposed method with four state-of-the-art community detection methods that include MOSES [9], DENGGRAPH [3], gSkeletonClu [10], and CFinder [5]. The evaluation is performed based on two scoring measures which include *omega index* and Normalized Mutual Information (NMI). Both Omega and NMI are generalized scoring measures used for evaluating both overlapping and non-overlapping community structures. gSkeletonClu and MOSES are parameter free methods and do not require an input. On the other hand, CFinder requires an input parameter k to define the clique size which is set to $k = 4$ in our experiment as it has been shown to generate best results for this clique size. For DENGGRAPH, the input parameters ε and μ are varied to generate the best possible results.

We present experimental results on a dynamic network dataset provided in [11], which comprises two weighted networks of face-to-face proximity between 242 individuals representing students and teachers in a primary school over a period of two days. The two networks correspond to two days of study wherein a daily contact network is provided. The nodes in this network represent students and teachers, and edges correspond to the interactions between them. The weight of an edge represents the number of times two nodes have interacted during the day. The students actually belong to ten different classes which can represent the ground truth communities. The teachers do not specifically belong to any class and interact with any student community.

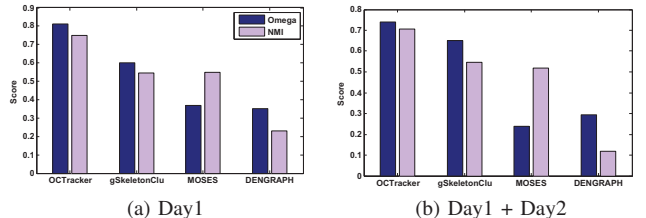


Fig. 1: Experimental results on a primary school dynamic network considering the network (a) after day-1 and (b) after day-2 (i.e., merged day-1 and day-2 network).

Figure 1a shows the comparison of performance scores (Omega and NMI) for the various methods on the interaction network of the individuals after day-1. The scores are computed against the known ground truth for day-1. In order to detect the evolutionary events, the community structure detected by OCTracker for day-1 forms its initial community state. This initial community structure is now adapted to changes in the network, i.e., by adding the interactions for day-2 to the underlying network which could also include adding new nodes, as discussed in section IV. Figure 2 shows the dynamic changes that occur in the community structure of the primary school interaction network over two

days as tracked by OCTracker. Initially on day-1 network, OCTracker detects 9 communities labeled as $A - H$, of which community C overlaps with D and E overlaps with I . Now, the interactions for day-2 are merged with the underlying day-1 network which leads to addition of some new nodes and edges, and increase in the weights of some already existing edges. OCTracker now scans the changes in the network as discussed in section IV and tracks the resulting community-centric changes in the initial community structure. As shown in figure 2, almost all the initial communities gain nodes resulting in their expansion. Two important evolutionary events are detected by OCTracker after the second day of interactions. Firstly, the two overlapping communities C and D merge to form a single community labeled as $C + D$. Secondly, community G splits into two overlapping communities labeled as G_1 and G_2 . Moreover, after the second day of interactions, many communities begin to overlap with each other which are represented by overlapping circles in figure 2.

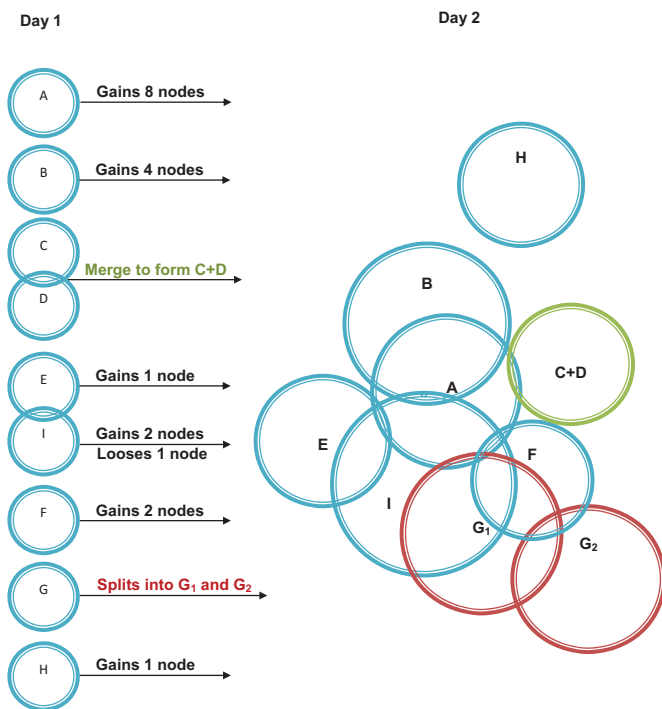


Fig. 2: Community evolution tracking in a primary school dynamic network.

Figure 1b shows the comparison of performance scores (Omega and NMI) for the various methods on the interaction network of the individuals after day-2, i.e., the network represented by merging the interactions and nodes for both day-1 and day-2. The scores are computed against the known ground truth for both day-1 and day-2 data. As can be seen from the results, OCTracker performs better than all other methods in question for the complete primary school interaction network over two days. To generate the results for OCTracker on the primary school network dataset, the input parameter η is set to 65%. Surprisingly, CFinder could not generate any results

for the primary school network data due to its higher space complexity.

VI. CONCLUSION

In this paper, we have proposed a novel density-based framework OCTracker that can track the evolution of overlapping community structures in online social networks. The novelty of the proposed framework lies in the approach for allowing the communities to overlap, and its distance function which is defined as a function of the average interactions between a node and its neighborhood. Furthermore, unlike other density based methods for which the neighborhood threshold is to be set by the users, which is generally difficult to determine, OCTracker computes a local neighborhood threshold for each node from the underlying network.

ACKNOWLEDGMENT

The authors would like to thank King Abdulaziz City for Science and Technology (KACST) and King Saud University for their support. This work has been financially supported by KACST under the NPST project number 11-INF1594-02.

REFERENCES

- [1] C. Tantipathananandh, T. Berger-Wolf, and D. Kempe, "A framework for community identification in dynamic social networks," in *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, ser. KDD '07. New York, NY, USA: ACM, 2007, pp. 717–726.
- [2] Y.-R. Lin, Y. Chi, S. Zhu, H. Sundaram, and B. L. Tseng, "Analyzing communities and their evolutions in dynamic social networks," *ACM Trans. Knowl. Discov. Data*, vol. 3, pp. 8:1–8:31, April 2009.
- [3] T. Falkowski, A. Barth, and M. Spiliopoulou, "DENGRAPH: a density-based community detection algorithm," in *Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence*. IEEE Computer Society, Washington, DC, USA, 2007, pp. 112–115.
- [4] M. Ester, H. Krieger, S. Jörg, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proceedings of the International Conference on Knowledge Discovery from Data*, 1996, pp. 226–231.
- [5] G. Palla, I. Derényi, I. Farkas, and T. Vicsek, "Uncovering the overlapping community structure of complex networks in nature and society," *Nature*, vol. 435, no. 7043, pp. 814–818, 2005.
- [6] G. Palla, A. I. Barabási, T. Vicsek, and B. Hungary, "Quantifying social group evolution," *Nature*, vol. 446, pp. 664–667, 2007.
- [7] D. Greene, D. Doyle, and P. Cunningham, "Tracking the evolution of communities in dynamic social networks," in *Proceedings of the 2010 International Conference on Advances in Social Networks Analysis and Mining*, ser. ASONAM '10. Washington, DC, USA: IEEE Computer Society, 2010, pp. 176–183.
- [8] T. Falkowski, A. Barth, and M. Spiliopoulou, "Studying community dynamics with an incremental graph mining algorithm," in *Proceedings of the 14th Americas Conference on Information Systems (AMCIS)*, 2008.
- [9] A. McDaid and N. Hurley, "Detecting highly overlapping communities with model-based overlapping seed expansion," in *Proceedings of the 2010 International Conference on Advances in Social Networks Analysis and Mining*, ser. ASONAM '10. Washington, DC, USA: IEEE Computer Society, 2010, pp. 112–119.
- [10] H. Sun, J. Huang, J. Han, H. Deng, P. Zhao, and B. Feng, "gskeletonclu: Density-based network clustering via structure-connected tree division or agglomeration," in *Proceedings of the 2010 IEEE International Conference on Data Mining*, ser. ICDM '10. Washington, DC, USA: IEEE Computer Society, 2010, pp. 481–490.
- [11] J. Stehl, N. Voirin, A. Barrat, C. Cattuto, L. Isella, J.-F. Pinton, M. Quaghiotto, W. V. den Broeck, C. Rgis, B. Lina, and P. Vanhems, "High-resolution measurements of face-to-face contact patterns in a primary school," *CoRR*, vol. abs/1109.1015, 2011.