A Novel Method for Vertex Clustering in Dynamic Networks

Devavrat Vivek Dabke[1](✉) and Olga Dorabiala[2]

[1] Princeton University, Princeton, USA
ddabke@princeton.edu
[2] University of Washington, Washington, USA
olgad400@uw.edu

Abstract. In this paper, we introduce *spatiotemporal graph k-means* (STGkM), a novel, unsupervised method to cluster vertices within a dynamic network. Drawing inspiration from traditional k-means, STGkM finds both short-term dynamic clusters and a "long-lived" partitioning of vertices within a network whose topology is evolving over time. We provide an exposition of the algorithm, illuminate its operation on synthetic data, and apply it to detect political parties from a dynamic network of voting data in the United States House of Representatives. One of the main advantages of STGkM is that it has only one required parameter, namely k; we therefore include an analysis of the range of this parameter and guidance on selecting its optimal value. We also give certain theoretical guarantees about the correctness of our algorithm.

Keywords: Vertex clustering · Dynamic networks · Graph clustering · Community detection · k-means

1 Introduction

Dynamic graphs are becoming increasingly prevalent mathematical structures as we collect more detailed data on the world around us. Though graphs have traditionally been studied as static objects, the dynamic setting better captures systems that evolve over time. Also called "time-varying" or spatiotemporal graphs, they extend static graphs by permitting edges to change over time, and they inherently reflect many systems, e.g., road networks, online communities, and epidemic spread. Since they are much less understood than their static counterparts, they pose an exciting and rich area of study.

Much of the literature on dynamic graphs focuses on extending well-known concepts from the static case like connectivity [15], optimal routing [6], induced dynamical systems [8], and more. Our work fits into this foundational literature by extending the notion of vertex clustering for the purpose of community

Both authors contributed equally to this work.
See https://github.com/dynestic/stgkm for the associated code for this project.
We would like to acknowledge J. Nathan Kutz (U. of Washington) for his support.

detection. Graph clustering is a fundamental tool for network analysis, with applications across the social and natural sciences, and we seek to bring this tool to the dynamic setting. In dynamic graph clustering, we find a partition of graph vertices that takes into account both spatial similarity—so that there are many edges within a cluster and relatively few between clusters—and temporal similarity, so clusters stay consistent over time. These partitions help us detect latent community structures.

In this paper, we propose a method we call *spatiotemporal graph k-means* (STGkM) that is able to track the multi-scale relationships between graph vertices. STGkM applies a two-phase clustering approach, wherein the first phase outputs an assignment for each vertex at every time step and the second phase produces a single, long-term partition of vertices based on historical cluster membership. STGkM identifies communities of interest and automatically tracks their evolution over time. To validate our method, we provide certain theoretical guarantees and showcase the utility of STGkM on synthetic and real-world datasets.

2 Related Work

In static graphs, vertex clustering has a broad literature with interdisciplinary interest and there has been a push to extend these results to the dynamic setting. Most approaches to dynamic community detection find clusters independently at each time step and then use aggregation to successively infer relationships between partitions [11]. These methods are often unable to achieve temporal smoothness and inevitably do not capture the dynamics of the network [12]. Another subset of methods first constructs a single coupling graph that summarizes the temporal properties of the dynamic network and then runs a classic community detection method on this graph [22]. As with aggregation, the use of coupling graphs results in a loss of temporal information.

Evolutionary clustering addresses this shortcoming through a unified framework, where clusters are iteratively formed based on current network structure and previous partitions. A cost function regulates the tradeoff between cluster quality at each snapshot and cluster consistency [4]. This framework has been successfully adopted and refined [5,19]. Other lines of research extend static community detection using online algorithms [23], machine learning [24], or systems-based approximation algorithms [9]. These papers leverage diverse methods to contend with the sometimes staggering size of dynamic networks.

Our method, STGkM, develops a unified framework akin to, but distinct from, evolutionary clustering [4]. STGkM, achieves temporal smoothness by restricting the search space of new cluster centers based on temporal reachability from previous centers. In addition, our method goes further than existing techniques to also extract long-lived communities of vertices based on historical dynamic cluster membership. STGkM is the graph analogue to our previously developed point-based method [10]. We first introduced a notion of STGkM in an extended abstract with some preliminary evidence of its effectiveness, but we have since refined our approach and this paper provides our complete results [7].

Finally, we note that there are numerous methods to group vertices within a dynamic network, each with its own motivation, challenges, and rich literature. Our method of vertex partitioning prioritizes long-term stable connections and our main theoretical result highlights the relationship to the distinct concept of connected components. However, there are many other interesting notions of connectivity [3] with variants in stochastic settings [1] along with other related problems, like motif detection [13], centrality measurement [2,14,18,25], and even novel frameworks for capturing properties of dynamic networks [17].

3 Spatiotemporal Graph *k*-means (STG*k*M)

Our goal is to partition a vertex set given a dynamic graph. In STGkM, we construct a partition by finding *central* nodes to represent each cluster and then assigning each remaining vertex based on its closest central node.[1] Just as with k-means, we define the problem of finding good clusters as a minimization problem; our novel objective has a unified formulation over space and time that predicts a partition for each vertex at every time step. After pre-processing, STGkM consists of two phases: in a single pass of Phase 1, the algorithm outputs vertex membership and dynamic cluster center journeys; in Phase 2, we extract the long-lived communities from the graph.

Setup

As input data, we need: a (finite) vertex set V, a (finite) time set $\mathbb{T} \subset \mathbb{N}$, a dynamic graph $\mathcal{G} = (V, E^t)_{t\in\mathbb{T}}$ where $E^t \in V \times V$, and an optional non-negative cost function $\omega^t : V \times V \to \mathbb{R}$ for all $t \in \mathbb{T}$. Our parameters are $k \in \mathbb{N}$, the number of clusters, and—optionally—$\lambda \in \mathbb{N}$, the maximum cluster center drift, and $\gamma \in \mathbb{Z}_{\geq 0}$, the drift time window.

Pre-processing

For all pairs of vertices across time, we compute and store the s-journey δ, see [15] for details. The value of $\delta^t(u, v)$ is the length of the shortest journey (i.e. dynamic path) starting at vertex u at time t and ending at vertex v. If no such journey exists, it assigns $+\infty$. Thought not a true metric (it is missing symmetry and coincidence), this function has the same purpose as a distance in classical k-means.[2] We also define the related true metric $\tilde{\delta}^t(u, v) \triangleq \delta^t(u, v) + \delta^t(v, u)$ with the additional convention that $\tilde{\delta}^t(u, u) = 0$.

Phase 1

Given a fixed value of k, the first phase of STGkM selects a set of k vertices to serve as cluster centers and assigns each vertex to a cluster at every time step.

[1] Our approach is perhaps more analogous to k-medoids, but in a network context, the distinction between k-means and k-medoids is not obvious.

[2] If no weight functions are provided or if the weight functions only output natural numbers, then δ will assign only natural numbers.

Vertices have the flexibility to switch cluster membership at every time step, but cluster centers are constrained by drift parameters λ and γ.

Natural Objective. The natural extension of k-means would be to optimize the objective function in Expression 1:

$$\min_{c \in \mathcal{C}, W \in \mathcal{W}} \sum_{t \in \mathbb{T}} \sum_{u \in V} \sum_{j \in [k]} W^t_{u,j} \cdot \tilde{\delta}^t(u, c^t_j) \tag{1}$$

where we minimize over cluster centers $\mathcal{C}$ and assignment tensors $\mathcal{W}$. Formally, $\mathcal{C}$ is the set of all sequences of length $|\mathbb{T}|$ where each element is an ordered subset of V with k elements; $\mathcal{W} \triangleq \{0,1\}^{|\mathbb{T}| \times |V| \times k}$ such that $\sum_{j \in [k]} W^t_{u,j} \geq 1$. Note that we allow vertices to belong to multiple clusters simultaneously, and each vertex is assigned to at least one cluster at every time step.

Objective with Regularization. Optimizing Expression 1 is NP-hard[3], so we instead iteratively optimize a modified objective function that restricts the search space. We begin by choosing initial cluster centers c^0 to be the nodes that are most closely connected to all others at t_0. When there are ties, we sample randomly. OlgAt each time t henceforth, we assume that we have chosen optimal cluster centers c^s for all $s < t$, and we minimize Expression 2.

$$\begin{gathered} \min_{c,W} \sum_{u \in V} \sum_{j \in [k]} W^t_{u,j} \cdot \delta^t(u, c^t_j) \\ \text{such that } \delta^{t-q}(c^{t-1}_j, c^t_j) \leq \lambda, \text{ where } 1 \leq q \leq \gamma \text{ and } 1 \leq j \leq k \end{gathered} \tag{2}$$

The constraint in Expression 2 imposes that the center of a given cluster can only switch from vertex u to vertex v if the distance between them is no more than λ for the previous γ time steps. This regularization serves two purposes: first, it associates dynamic clusters between time steps; second, it restricts the search space for cluster centers[4]. As we decrease λ or increase γ, we decrease the number of potential centers at time t and enforce stricter cluster consistency; see Fig. 1 for an example.

In practice, we update the center of a cluster only if the objective is improved. When we encounter the case where selecting new cluster centers is infeasible, we update clusters individually instead of jointly. Our algorithm terminates until either the clusters stabilize or we reach a maximum number of iterations.

[3] To see why, observe that k-medoids is NP-hard [20].

[4] In the worst case, e.g. when the graph is complete at every time step, optimizing this objective is still NP-hard, but in practice, it makes STGkM tractable.

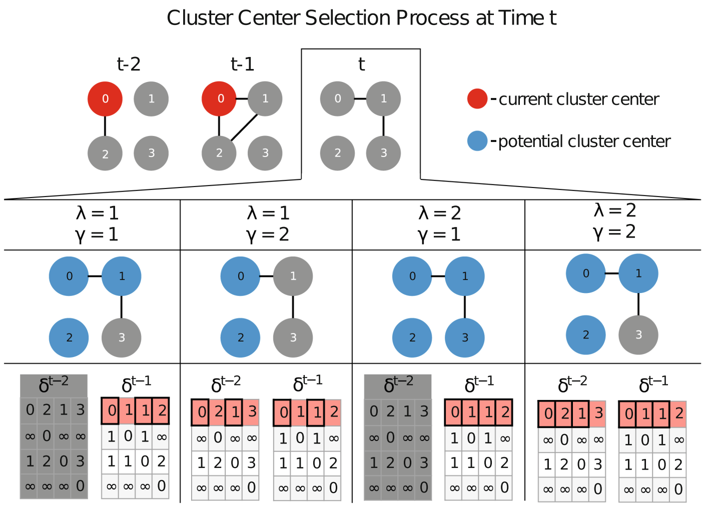


Fig. 1. At time t, c_0^t is chosen based on c_0^{t-1}. The drift time window γ determines for how many previous time steps centers must be within maximum drift λ of one another. The objective in Expression 2 is evaluated for all potential cluster centers; the center that minimizes the objective is chosen.

Phase 2

By building on Phase 1, Phase 2 of STGkM aims to identify the long-lived partitions of graph vertices. The output is an assignment of communities containing vertices with the most similar spatiotemporal characteristics. Intuitively, we expect vertices with similar partitioning histories to belong to the same persisting community in the long run.

Recall that the *Hamming distance* is defined by counting the number of entries where two matrices disagree: $H(u,v) \triangleq |\{(t,k) : W_{u,k}^t \neq W_{v,k}^t\}|$. Using this distance, we define similarity sim(u,v) as

$$\mathrm{sim}(u,v) \triangleq 1 - \frac{H(u,v)}{|\mathbb{T}|} \tag{3}$$

This definition gives us a powerful way to compare all pairs of vertices. Since sim$(\cdot,\cdot)$ is compatible with traditional clustering techniques, we input it to agglomerative clustering. We then output the resulting partition to get a clustering of the vertices based on long-lived communities as desired.

4 Results

4.1 Algorithmic Analysis

The main feasibility issue with STGkM arises from finding new cluster centers at every time step. Evaluating all possible subsets of size k of $|V|$ is NP-hard.

The objective in Expression 2 does not obviate this possibility in the worst case, even though in practice, the added regularization results in a sufficiently fast algorithm. There are other strategies we could deploy to minimize our objective (e.g. greedy algorithms, subsampling, further constraints on cluster center drift) that may be theoretically efficient, but we did not explore these possibilities in detail, since our chosen strategy works in practice. Also, we do not provide formal guarantees on approximation quality for Expression 2.

4.2 Connected Components

Although the clusters that STGkM generates are distinct from connected components, we can find connected components under certain conditions, as presented in Theorem 1. Though our method may not be most efficient way to find connected components (and there are other notions of connected components), our theoretical result provides evidence that STGkM can find interesting partitions.

Definition 1 (Dynamic Connected Component). *Vertices u, v are* (dynamically) connected *if there exists a finite journey from u to v and from v to u over all time steps. A set of vertices U (where $U \subseteq V$) is a (dynamic) connected component if all vertices in this set are connected and there is no vertex in $V \setminus U$ that is connected to a vertex in U.*

Lemma 1. *For two vertices u, v in distinct connected components, there exists some time step t_0 such that $\tilde{\delta}^{t_0}(u, v) = \infty$.*

Proof. By definition, if u, v are not connected, then there must be some time step t_0 at which there is no finite journey from u to v or v to u. $\square$

Definition 2 (Self-Connected). *A dynamic network is* self-connected *if each vertex is connected to itself.*

Lemma 2. *In a self-connected dynamic network, connectivity is an equivalence relation and the connected components are the respective equivalence classes.*

Proof. By construction, connectivity is symmetric. By self-connection, connectivity is reflexive. Connectivity is also transitive: if vertex u is connected to v and v is connected to w, then there exists a journey from u to w via v for all time steps. By construction, each connected component only contains vertices that are connected and no other vertices, so it is an equivalence class. $\square$

We now make use of two further concepts: a *non-stranding* dynamic graph is one where for every time step, every vertex has at least one edge, and *holding*, where each vertex has a self-loop at every time step [15]. We introduce Lemma 3, which is related to (but distinct from) Proposition 3.5 in [15].

Lemma 3. *For two vertices u, v in distinct connected components in a non-stranding, holding dynamic network, there exists some time step t_0 such that $\tilde{\delta}^t(u, v) = \infty \ \forall \ t > t_0$.*

Proof. We proceed by induction. For our base case, by Lemma 1, there exists t_0 such that $\tilde{\delta}^{t_0}(u,v)=\infty$. Therefore, either $\delta^{t_0}(u,v)=\infty$ or $\delta^{t_0}(v,u)=\infty$, so (without loss of generality) we assume $\delta^{t_0}(u,v)=\infty$.

For our inductive step, we will show that if for some t that $\delta^t(u,v)=\infty$, then $\delta^{t+1}(u,v)=\infty$. By holding $\delta^t(u,u)=1$ and so $\delta^t(u,v)\leq 1+\delta^{t+1}(u,v)$, which immediately implies that $\delta^{t+1}(u,v)=\infty$. By induction, for all $t>t_0$, $\delta^t(u,v)=\infty$ and so $\tilde{\delta}^t(u,v)=\infty$ as desired. □

Theorem 1. *Given a holding, non-stranding dynamic graph with k connected components, the partition of vertices induced by the optimal solution to Expression 1 is exactly the connected components given sufficient time.*

Proof. For every pair of vertices u,v that are not connected, there exists some time step such that $\tilde{\delta}^t(u,v)=\infty$ for all $t>t_{u,v}$ by Lemma 3. For a pair of vertices u,v, denote $t_{u,v}$ to be the minimum such time step if they are disconnected or 0 otherwise. Let $t^*=\max\{t_{u,v}\}$. This notion is well-defined because every vertex is in a connected component at least with itself by Lemma 2.

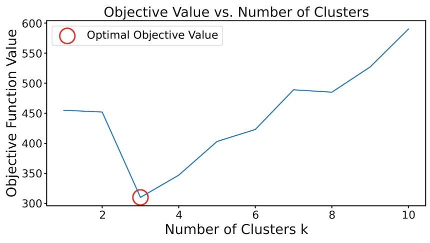
For all t such that $t\geq t^*$, note that $\tilde{\delta}^t(u,v)=\infty$ if and only if u,v are in different connected components. If two entries in c^t are in the same connected component, then there must be one vertex v that is not connected to any cluster center in c^t by the pigeonhole principle and thus $\tilde{\delta}^t(v,c^t_j)=\infty$ for all j. At least one entry in $W^t_{v,\bullet}$ must be 1 by construction and thus

$$\sum_{j\in[k]} W^t_{v,j}\cdot\tilde{\delta}^t(u,c^t_j)=\infty$$

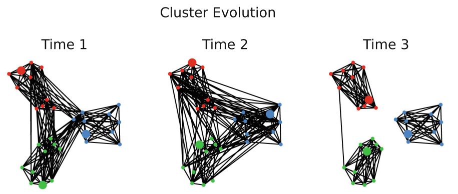
Conversely, we can select vertices c_j with $j\in[k]$ such that each is in a distinct connected component. Now, construct W^t such that $W^t_{u,j}$ is 1 when u and c_j are connected and is 0 otherwise. With this construction, the sum below is finite:

$$\sum_{u\in V}\sum_{j\in[k]} W^t_{u,j}\cdot\tilde{\delta}^t(u,c^t_j)$$

and this constructed W^t is optimal. For two connected vertices u,v, $W^t_{u,\bullet}=W^t_{v,\bullet}$ so $\text{sim}(u,v)\geq 1-\epsilon$ where $\epsilon=\frac{t^*\times k}{|\mathbb{T}|}$. For two disconnected vertices x,w, there exists at least one index j such that $W^t_{x,j}\neq W^t_{w,j}$ so $\text{sim}(x,w)<\epsilon+\frac{k-1}{k}$. With sufficiently large $|\mathbb{T}|$, we will correctly separate these clusters. □



(a) Objective function value versus number of clusters on a synthetic dataset.



(b) Three snapshots of a dynamic graph and the dynamic clustering as predicted by STGkM. Cluster centroids are identified by enlarged nodes.

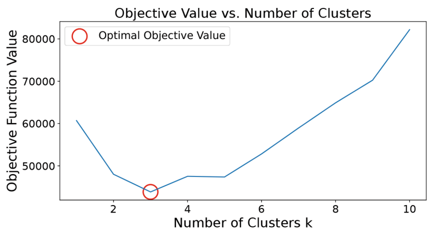
Fig. 2. STGkM on a synthetic dataset, consisting of three ground-truth clusters.

4.3 Experimental Insights

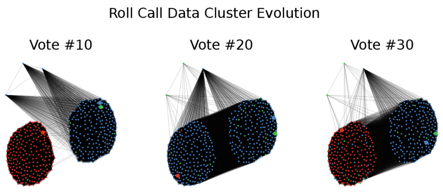
Choosing *k*. Perhaps the greatest challenge in using a k-means-based approach for clustering is determining the optimal number of clusters k. With regard to classical k-means, one of the most common methods for choosing k is the Elbow Method [16], wherein the sum of square error of each cluster is calculated, and the value of k which results in the most extreme difference (the elbow) is chosen. We employ an approach similar to the Elbow Method for STGkM.

We calculate the value of Expression 2 for every value of k being considered, and seek the value of k for which the objective is minimized. It is important to note, that as opposed to the Elbow Method, which finds the most extreme difference in objective values, we seek the minimal value. The reason for this is that we allow for vertices to belong to multiple clusters simultaneously. Unlike classical k-means where increasing k results in points getting progressively closer to their centers, in STGkM, increasing k is likely to cause vertices to be assigned to progressively more clusters simultaneously. Consider the case of n vertices in a clique: cluster centers will be assigned only to their own cluster, whereas every other vertex will be assigned to all clusters. The value of Expression 2 is thus $(n-k)k$. Restricting to small k, the objective value is minimized when $k=1$ and increases as k increases. In a clique, we would expect $k=1$.

Synthetic Data. We begin by applying STGkM to a synthetic dataset, consisting of three clusters with 10 fully connected nodes in each cluster, tracked over 20 time steps. The result is a dynamic graph with 30 nodes and 300 edges at each time t. At every time step we randomly choose up to 30 edges to remove within clusters and up to 30 edges to add between clusters. We run STGkM with $\lambda=1$, $\gamma=1$. Following the method for choosing k described previously, we set $k=3$. The selection process for choosing k is shown in Fig. 2a, and a snapshot of the evolution of detected clusters is shown in Fig. 2b. As expected, three communities persist throughout the duration of the simulation.



(a) Objective function value versus number of clusters on the Roll Call dataset.

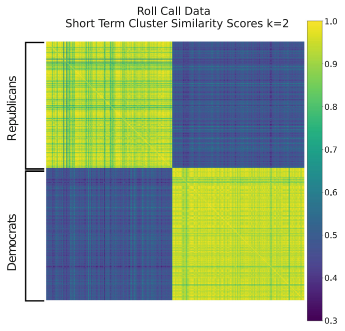


(b) Snapshots of the dynamic graph created by the Roll Call dataset. Cluster centroids are identified by enlarged nodes.

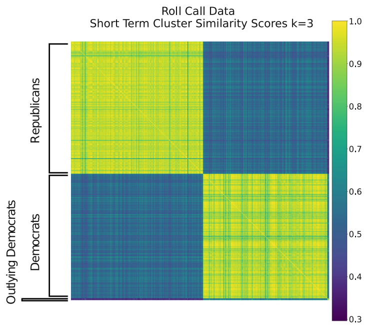
Fig. 3. STGkM on the Roll Call dataset.

Detecting Political Parties. To demonstrate the utility of STGkM on a larger, real-world dataset, we turn to politics. Communities naturally arise is politics, particularly in recent years where we have witnessed polarization with political figures consistently voting along party lines. Taking inspiration from [21], we form a dynamic graph based on 100 roll call votes from the House of Representatives between June 21, 2023 and July 27, 2023. Each vote is a time step, each representative is a node, and nodes are connected if they vote the same way on a bill. Possible votes are "Yea", "Nay", and "Present". If a representative does not cast a vote, they have no connecting edges for that vote. The ground truth communities are representative's affiliated political parties. By running STGkM on the roll call graph, we can identify the communities of representatives that vote similarly and observe how those communities evolve over time.

We choose our maximum center drift to be $\lambda = 1$ and our time connectivity to be $\gamma = 5$. Intuitively, we expect $k = 2$, but our k selection process recommends $k = 3$, as seen in Fig. 3a. We find that when we run STGkM with $k = 2$, we correctly separate Democrats and Republicans in our long term clusters, but interestingly enough, when we run STGkM with $k = 3$, we additionally find a sub-community of three Democrats. Upon further investigation, we find that these three Democrats, Rep. Joaquin Castro, Rep. Emanuel Cleaver, and Rep. Michael Kelly, very often vote "Present" together, as opposed to the majority Democratic party vote. The evolution of our dynamic clusters is visualized in Fig. 3b. We observe two large persistent clusters in red and blue. We also see how in some votes, such as #20 where most representatives vote identically, almost all nodes are assigned to one cluster. The third, green cluster often contains the three outlying Democrats, as well as other "Present" voters over various votes. Figures 4a and 4b visualize the similarity scores, as defined in Eq. 3, between the cluster assignment histories for each pair of representatives. The rows and columns of the similarity matrix are ordered according to the long-term communities discovered by STGkM. In Fig. 4a, these clusters correspond to Republicans followed by Democrats, while in Fig. 4b, the three outlying Democrats are moved



(a) Short term clustering similarity between nodes in the Roll Call dataset using STGkM with $\gamma = 5$, $\lambda = 1$, and $k = 2$.



(b) Short term clustering similarity between nodes in the Roll Call dataset using STGkM with $\gamma = 5$, $\lambda = 1$, and $k = 3$.

Fig. 4. Similarity matrices of the short term clustering similarity between nodes in the Roll Call dataset. Rows and columns of the matrices are organized by detected long-term community membership.

to the final three rows and columns of the matrix. We observe a distinct color difference between these three rows and the remainder of the matrix, demonstrating that the similarity between the outlying Democrats and remaining Democrats is much lower.

5 Conclusion

We introduce spatiotemporal graph k-means (STGkM) for community detection by vertex clustering on dynamic graphs. This approach is unified over space and time and gives us the ability to analyze both the short- and long-term partitions of graph vertices, monitor the multi-scale relationships between communities, and has just three explainable parameters, only one of which is required. We provide a principled approach to estimating the required parameter: the number of clusters k. We also state some theoretical guarantees that explain clustering behavior under certain conditions. Finally, we carry out experiments on both a synthetic and real world dataset to empirically validate STGkM.

In our future work, we seek to improve the efficiency of STGkM, both in practice and in theory. As STGkM is applied to larger datasets, further approximation strategies will be necessary to ensure feasibility. We would like to provide guidance on the quality and convergence of our approximation strategies. The theoretical guarantees in this paper are only correct under narrow conditions, so we would like to provide more contexts in which clustering is assured to work correctly. We will also explore online extensions of STGkM, where we explore dynamic graphs in real-time. Finally, we seek a characterization of the expected properties of STGkM in a stochastic setting with the presence of noise.

References

1. Becker, R., et al.: Giant components in random temporal graphs. arXiv preprint arXiv:2205.14888 (2022)
2. Bergamini, E., Meyerhenke, H.: Approximating betweenness centrality in fully dynamic networks. Internet Math. **12**(5), 281–314 (2016)
3. Casteigts, A., Flocchini, P., Quattrociocchi, W., Santoro, N.: Time-varying graphs and dynamic networks. Int. J. Parallel Emergent Distrib. Syst. **27**(5), 387–408 (2012)
4. Chakrabarti, D., Kumar, R., Tomkins, A.: Evolutionary clustering. In: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 554–560 (2006)
5. Chi, Y., Song, X., Zhou, D., Hino, K., Tseng, B.L.: Evolutionary spectral clustering by incorporating temporal smoothness. In: Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 153–162 (2007)
6. Cleveland, J., et al.: Introducing tropical geometric approaches to delay tolerant networking optimization. In: 2022 IEEE Aerospace Conference (AERO), pp. 1–11 (2022)
7. Dabke, D.V., Dorabiala, O.: Spatiotemporal graph k-means. In: Proceedings of the Communities in Networks ComNets @ NetSci 2023 (2023)
8. Dabke, D.V., Karntikoon, K., Aluru, C., Singh, M., Chazelle, B.: Network-augmented compartmental models to track asymptomatic disease spread. Bioinform. Adv. **3**, vbad082 (2023)
9. DiTursi, D.J., Ghosh, G., Bogdanov, P.: Local community detection in dynamic networks. In: 2017 IEEE International Conference on Data Mining (ICDM), pp. 847–852 (2017)
10. Dorabiala, O., Webster, J., Kutz, N., Aravkin, A.: Spatiotemporal k-means. arXiv preprint arXiv:2211.05337 (2022)
11. Fortunato, S.: Community detection in graphs. Phys. Rep. **486**(3–5), 75–174 (2010)
12. Görke, R., Maillard, P., Schumm, A., Staudt, C., Wagner, D.: Dynamic graph clustering combining modularity and smoothness. J. Exp. Algorithmics **18**, 1–1 (2013)
13. Gurukar, S., Ranu, S., Ravindran, B.: Commit: a scalable approach to mining communication motifs from dynamic networks. In: Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data, SIGMOD 2015, pp. 475–489. Association for Computing Machinery, New York (2015)
14. Habiba, C.T., Tanya, Y.: Berger-Wolf. Betweenness centrality measure in dynamic networks, Technical Report 19, DIMACS (2007)
15. Hylton, A., et al.: A survey of mathematical structures for lunar networks. In: 2022 IEEE Aerospace Conference (AERO), pp. 1–17 (2022)
16. Kodinariya, T.M., Makwana, P.R., et al.: Review on determining number of cluster in k-means clustering. Int. J. **1**(6), 90–95 (2013)
17. Latapy, M., Viard, T., Magnien, C.: Stream graphs and link streams for the modeling of interactions over time. Soc. Netw. Anal. Min. **8**(1), 61 (2018)
18. Lerman, K., Ghosh, R., Kang, J.H.: Centrality metric for dynamic networks. In: Proceedings of the Eighth Workshop on Mining and Learning with Graphs, MLG 2010, pp. 70—77. Association for Computing Machinery, New York (2010)
19. Lin, Y.-R., Chi, Y., Zhu, S., Sundaram, H., Tseng, B.L.: FacetNet: a framework for analyzing communities and their evolutions in dynamic networks. In: Proceedings of the 17th International Conference on World Wide Web, pp. 685–694 (2008)

20. Megiddo, N., Supowit, K.J.: On the complexity of some common geometric location problems. SIAM J. Comput. **13**(1), 182–196 (1984)
21. Reda, K., Tantipathananandh, C., Johnson, A., Leigh, J., Berger-Wolf, T.: Visualizing the evolution of community structures in dynamic social networks. In: Computer Graphics Forum, vol. 30, pp. 1061–1070. Wiley Online Library (2011)
22. Rossetti, G., Cazabet, R.: Community discovery in dynamic networks: a survey. ACM Comput. Surv. **51**(2), 1–37 (2018)
23. Ruan, B., Gan, J., Wu, H., Wirth, A.: Dynamic structural clustering on graphs. In: Proceedings of the 2021 International Conference on Management of Data, pp. 1491–1503 (2021)
24. Yao, Y., Joe-Wong, C.: Interpretable clustering on dynamic graphs with recurrent graph neural networks. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 35, pp. 4608–4616 (2021)
25. Yen, C.-C., Yeh, M.-Y., Chen, M.-S.: An efficient approach to updating closeness centrality and average path length in dynamic networks. In: 2013 IEEE 13th International Conference on Data Mining, pp. 867–876 (2013)