Vertex clustering in diverse dynamic networks

Devavrat Vivek Dabke[1†], Olga Dorabiala[2†]

1 Level Ventures, New York, NY, USA
2 Dept. of Applied Mathematics, University of Washington, Seattle, WA, USA

† These authors contributed equally to this work.
* olgad400@uw.edu

Abstract

In this paper, we provide additional theoretical and experimental results for *spatiotemporal graph k-means* (STGkM)—a new unsupervised method to cluster vertices within a dynamic network. STGkM finds both short-term dynamic clusters and a "long-lived" partitioning of vertices within a network whose topology is evolving over time; we first introduced this technique in a recent conference paper. In this work, we provide an updated exposition of the algorithm with a more efficient relaxation scheme. We review our previous theoretical results with respect to connected components, along with further analysis for certain important classes of dynamic networks to distinguish STGkM from connected components and static clustering. We also provide results for the stochastic setting for the first time. With respect to empirical results, we report on additional experiments from the previously used United States House of Representatives dataset, along with new results on a dynamic scientific citation network and Reddit dataset. These findings demonstrate that STGkM is efficient, informative, and operates well in diverse settings. Finally, as previously noted, one of the main advantages of STGkM is that it has only one required parameter: k, the number of clusters; we therefore include an extended analysis of the range of this parameter and guidance on selecting its optimal value.

Author summary

With the explosion of data about the world around us, we must constantly develop new ways of studying datasets. One popular method for analysis is k-means, which can identify clusters of related objects based on shared characteristics. Traditionally, k-means worked over sets of objects (e.g. animal subjects in a biology study) for which it was possible to define a list of consistent, numerical, and complete "features" or pieces of information (like age or weight). Over the years, this algorithm has been adapted for a variety of datasets and implemented in efficient code libraries. Our paper builds on this vast literature to extend k-means to the setting of networks that change over time, and we provide a practical and efficient implementation of it for real-world usage. We show that our new algorithm works on datasets like the United States House of Representatives roll call votes, citation networks from major publications, and a sample of Reddit posts. We also provide formal mathematical proofs and demonstrate the theoretical soundness of our technique.

1 Introduction

As we have previously noted [1, 2], dynamic graphs are prevalent mathematical structures that capture important aspects of the rich and frenetic world around us. Though graphs, or networks, have traditionally been studied as static objects, many such systems evolve over time. Also called "time-varying" or spatiotemporal graphs, they extend static graphs by permitting edges to change, and they inherently reflect many systems, e.g., road networks, online communities, and epidemic spread. While we have made strides in understanding them, there are still many exciting open questions about dynamic graphs, especially when compared to our relatively compendious knowledge about static ones.

In this paper, we expand upon *spatiotemporal graph k-means* (STGkM)—a novel technique that we first introduced as a recent conference paper [1]—which 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.

Leveraging some theoretical analysis, we can show when and how our method relates to dynamic connected components. In some cases, we exactly align with this technique for vertex clustering, but we also demonstrate where our method finds sensible clusters when connected components is too strong of a concept. We also demonstrate why dynamic clustering is superior to simple static clustering in a dynamic network and we provide a basic result demonstrating robustness to noise. Our empirical results show, first and foremost, that STGkM is applicable to a diverse range of datasets. It is also efficient and tractable. We also observe many interesting results, like finding political parties in voting networks and evidence for polarization in online discourse: here, we provide mathematical evidence for corresponding results in the social sciences [3].

2 Related Work

Much of the literature on dynamic graphs focuses on extending well-known concepts from the static case like connectivity [4], optimal routing [5], induced dynamical systems [6], and more. Further work has been done in applying dynamic networks to sports analytics, machine learning [7], and epidemiological spread [8, 9]. Our work fits into this foundational literature by extending the notion of vertex clustering. 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. These partitions help us detect latent community structures.

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 [10]. These methods are often unable to achieve temporal smoothness and inevitably do not capture the dynamics of the network [11]. 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 [12]. 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 [13]. This framework has been successfully adopted and refined [14,15]. Other lines of research extend static community detection using online algorithms [16], machine learning [17], or systems-based approximation algorithms [18]. 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 [13]. 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 [19]. We first introduced a notion of STGkM in an extended abstract with some preliminary evidence of its effectiveness and later as a conference paper, but we have since refined our approach and this work provides our complete results [1,2].

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 [20] with variants in stochastic settings [21] along with other related problems, like motif detection [22], centrality measurement [23–26], and even novel frameworks for capturing properties of dynamic networks [27].

3 Spatiotemporal Graph k-means (STGkM)

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.

3.1 Setup

As input data, we need:

- A (finite[2]) indexed vertex set: V
- A (finite) totally ordered time set: $\mathbb{T}$, usually $\mathbb{T} \subset \mathbb{N}$
- A dynamic graph: $\mathcal{G} = (V, E^t)_{t\in\mathbb{T}}$ where $E^t \in V \times V$
- (optional) a non-negative real cost function $\omega^t \colon V \times V \to \mathbb{R}$ 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.

[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 clear.

[2] In principle, SGTkM would exist in the infinite case, but we could not find a practical application and so will not consider this possibility in this paper.

3.2 Pre-processing

For all pairs of vertices across time, we compute and store the s-journey δ, see [4] 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. If we have a weight function, we consider this the "cost" of an edge, i.e., an edge with a smaller weight is a "shorter" edge and a missing edge has essentially "infinite" weight. If no weight functions are provided or if the weight functions only output natural numbers, then δ will assign only natural numbers.[3] 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 $\tilde{\delta}^t(u,u) = 0$.

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

3.3.1 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 each vertex is assigned to at least one cluster at every time step.

3.3.2 Objective with Regularization

Optimizing Expression 1 is NP-hard[4], 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. At 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

[3] As a practical matter, we can deploy a tie-breaking scheme when comparing two distances that are both infinity.

[4] To see why, observe that k-medoids is NP-hard [28].

centers[5]. As we decrease λ or increase γ, we decrease the number of potential centers at time t and enforce stricter cluster consistency; see Figure 1 for an example.

3.3.3 Optimizing Our Objective

We can further make our method more efficient, see Section 5.3.1 for empirical performance comparisons. In optimizing Expression 2, observe that at each time step, each vertex could belong to multiple clusters, since the rows of W^t are not restricted; we only require the overall tensor to be binary. However, we can further require that each row of W^t have exactly one non-zero entry; in other words, each vertex can only be in one cluster. The former approach, as explored in [1], allows us to avoid making strong assignments and potentially falling into a local minimum, but increases computational complexity. The latter approach is more attractive for the analysis of larger graphs, as it restricts the search space of potential cluster centers even further. We therefore focus on the latter method in this paper so that we can explore larger dynamic graph structures.

Our algorithm is an adaptation of FasterPAM with eager swapping, as detailed in [29]. FasterPAM consists of two phases: the BUILD phase, which finds initial cluster centers, and the SWAP phase, which efficiently seeks iterative improvements for the cluster centers by scanning over all vertices in the graph. We run the BUILD phase, as implemented in [29], on the first time slice of our matrix storing s-journeys. For all following time steps, we update the center of a cluster only if the objective value is improved. To do so, we run eager SWAP phases, where instead of considering all points as potential swaps for each center, we consider only those points which are reachable based on the constraint in Expression 2. We greedily update cluster centers, performing any update that yields some improvement in the loss function. Our algorithm terminates until either the clusters stabilize or we reach a maximum number of iterations.

3.3.4 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 makes it unlikely. In order to further improve run time, we build upon FasterPAM, as discussed above and take advantage of the fact that FasterPAM deploys the strategies of local search and greedy updates. As discussed in [29], the runtime complexity of a single BUILD phase of FasterPAM is $O(k|V|^2)$, and in the worst case, where every vertex would be reachable from every medoid, the runtime complexity of each SWAP phase would be $O(|V|(|V|-k))$. This means that in the worst case, Phase 1 of STGkM has a runtime complexity of $O(|\mathbb{T}||V|^2)$.

3.4 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: $H(u,v) \triangleq |\{(t,k) : W^t_{u,k} \neq W^t_{v,k}\}|$. Using this distance, we define similarity $\mathrm{sim}(u,v)$ as

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

[5] 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.

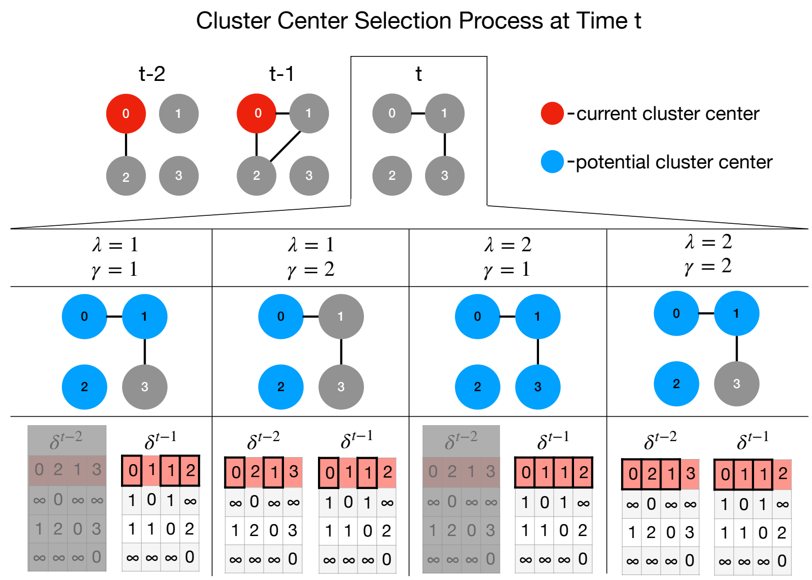

Fig 1. At time t, c_0^t is chosen based on c_0^{t-1}. The drift time window γ determines for how many previous time steps centers must be within maximum drift λ of one another. The objective in Expression 2 is evaluated for all potential cluster centers; the center that minimizes the objective is chosen.

This definition gives us a powerful way to compare all pairs of vertices. Since $\mathrm{sim}(\cdot,\cdot)$ 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 Theoretical Results

We provide some insight into how STGkM operates with some theoretical observations. Namely, our results are as follows:

1. Theorem 4.1: in certain cases, we find dynamic connected components. The key point is that we can recover a well-known concept of clusters. We provide sufficient conditions for when STGkM is identical to the problem of finding dynamic connected components. This result was first shown in [1] and we simply restate the main theorem here.
2. Theorem 4.3: in other cases, we recover something other than dynamic connected components, which both draws a distinction between the clusters induced by STGkM and dynamic connected components, as well as highlights the utility of STGkM when dynamic connected components is too strong for the task at hand.
3. Theorem 4.8: we demonstrate that our dynamic clustering method finds the "correct" number of clusters, while static clustering cannot do so by Lemma 4.7. In other words, we demonstrate where dynamic clustering with a unified spatio-temporal objective is superior to simpler static methods.
4. Theorem 4.11: we demonstrate that our technique is robust to noise and is correct in expectation. Admittedly, this result is relatively simple and we do not provide a sensitivity analysis with respect to noise.

4.1 Harmony with 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 4.1; this result, with more extensive proofs and intermediate results, is simply reproduced from [1] for context. 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 4.1 (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.*

Theorem 4.1. *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. See [1]. $\square$

4.2 More Granularity than Connected Components

Dynamic connectivity is a relatively strong requirement for two vertices. Indeed, if two vertices become merely disconnected for just one time step, they cannot be dynamically connected. With little noise (especially in real-world data), it is easy to generate dynamic networks with no true non-trivial dynamic connected components; alternatively, it is just as easy to find real-world data where the whole graph is one dynamic connected component and we cannot distinguish between vertices. To demonstrate this issue and how STGkM can handle such cases, we construct a special type of dynamic network, see Definition 4.2. We then show that it has only one dynamic connected component, but an intuitive notion of clusters. Lemma 4.2 shows that using dynamic connected components would fail. Theorem 4.3 demonstrates our ability to find the correct k clusters.

Definition 4.2 (Clique-cross-Clique)**.** *Let k be the "ground-truth" number of clusters and n be the number of vertices per cluster. We have a total of $N \triangleq k \times n$ vertices in V. For ease of indexing, we let $v_{i,j}$ be the j^{th} vertex in the i^{th} cluster, i.e. $i \in [k], j \in [n]$. We construct the following dynamic graph: $(v_{i,j}, v_{i',j'})$ is in E^t if and only if one of the following conditions is satisfied:*

1. *self-loops, i.e. $i = i', j = j'$*
2. *the vertices are in the same cluster, i.e. $i = i', j \neq j'$*
3. *the vertices have the same position in their respective clusters and it is their "turn" to be connected, i.e. $i \neq i', j = j'$ and $j = t \mod n$.*

We will call such a dynamic graph $\mathcal{G} = (V, E^t)$ a Clique-cross-Clique.

Lemma 4.2. *A Clique-cross-Clique has one dynamic connected component, namely all of its vertices.*

Proof. This dynamic graph is holding and non-stranding; at every time step, the static graph induced by $G^t = (V, E^t)$ is connected. Therefore, by Proposition 3.4 in [4], this graph is dynamically connected. Graphs that are dynamically connected have, by definition, one dynamic connected component. We note that intuitively, the idea is simply that each of the k clusters is fully connected at all time steps and that between any two clusters, there is always an edge. $\square$

Theorem 4.3. *Given a Clique-cross-Clique with k clusters, the partition of vertices induced by the optimal solution to Expression 1 is exactly the k clusters with sufficient time.*

Proof. First, we will show the simple fact that vertices in the same cluster have a smaller objective value than those in different clusters given sufficient time. We observe that $\delta^t(u,v) = 1$ for u, v in the same cluster for all time steps. If u, v are in different clusters, then $\delta^t(u,v) \geq 2$ at least for $\frac{k-1}{k} * T$ time steps (this bound is conservative) given sufficiently large T (specifically, $T > 2k$). Therefore, for any $W^t_{u,j} > 0$ that $W^t_{u,j} \cdot \delta^t(u,v) < W^t_{u,j} \cdot \delta^t(u,v')$ where u, v are in the same cluster but u, v' are in distinct clusters.

Second, we will show that the optimal cluster assignment is always to assign one entry of c^t to each cluster; we proceed by contradiction. Assume that we have found an optimal assignment c^t, W; select $j \in [k]$ and assume that c^t contains no vertices that are in cluster j; let v be some vertex in this cluster. By the pigeonhole principle, there must be two vertices in c^t that are in the same cluster: call them u, u' and let i, i' be their respective indices. We will show that we can switch one of these "duplicate" vertices to v to decrease our objective value. Select some vertex w in the same cluster as u and w' in the same cluster as v. Since $\delta^t(w,u) = \delta^t(w,u')$, we know that $(W^t_{w,i} + W^t_{w,i'}) \cdot \delta^t(w,u) = W^t_{w,i} \cdot \delta^t(w,u) + W^t_{w,i'} \cdot \delta^t(w,u')$. In other words, if we remove u' from $c^t_{i'}$, then we could simply update $W^t_{w,i}$ to take the value of $W^t_{w,i} + W^t_{w,i'}$ without changing the objective value. Thus, if we set $c^t_{i'}$ to v, then we observe that the overall objective value has not changed for vertices in the same cluster as u.

However, if we set $c^t_{i'}$ to v, by the fact that we showed in the beginning of this proof, there would be an update to W that would indeed decrease the overall objective value. For completeness, we observe that for a vertex that is not in the cluster with u nor with v would remain unaffected. Such an assignment is thus not minimal, which is a contradiction. $\square$

Remark 4.4. *Note that when $\lambda = \gamma = 1$, then Expression 2 and, in fact, even our single membership scheme also finds the optimal result.*

4.3 More Powerful than Static Clustering

One common clustering technique with dynamic networks is to simply convert a dynamic network into a static one. In particular, one easy way to "aggregate" a dynamic network is to count the total number of dynamic edges between two vertices. We can then perform traditional static clustering on such a static graph. However, this method will certainly miss important dynamic properties. We construct an example in Definition 4.3 and show that the total number of edges between vertices is uniform in Lemma 4.7—and so static clustering will not be able to meaningfully distinguish between vertices—but that STGkM will find the appropriate number of clusters, see Theorem 4.8. This example is, in fact, a motivating example for STGkM and provides some insight into how we can find *induced* clusters over time.

Definition 4.3 (Theseus Clique)**.** *Let U be the set of vertices $u_0, \ldots, u_{n-1}$ and W be the set of vertices $w_0, \ldots, w_{n-1}$. In this case, $V \triangleq U \cup W$ is our vertex set with $2n$ vertices. Let $r \in \mathbb{N}$ be the "round" index and $i \in \mathbb{N}$ be the "slice" index; we parameterizes our "time" index as $t(r,i) = r * n + (i \mod n)$.*

We thus define a dynamic graph $\mathcal{G} = (V, E^t)$ where

$$E^0 \triangleq \{(u,u') : u, u' \in U\} \cup \{(w,w') : w, w' \in W\}$$

and then

$$E^t \triangleq S^{r,i}(E^{t-1})$$

where $S^{r,i}$ *swaps all edges involving the vertices* $u_i, w_{(i+r) \bmod n}$. *Formally,* $(a,b) \in S^{r,i}(E)$ *if and only if:*

1. $a \notin P^{r,i}, b \notin P^{r,i}$ *and* $(a,b) \in E$
2. $a \in P^{r,i}, b \in P^{r,i}$ *and* $(a,b) \in E$
3. $a \in P^{r,i}, b \notin P^{r,i}$ *and* $(\overline{a},b) \in E$
4. $a \notin P^{r,i}, b \in P^{r,i}$ *and* $(a,\overline{b}) \in E$

where $P^{r,i} = \{u_i, w_{(i+r) \bmod n}\}$ *and* $\overline{\bullet}$ *of an element* $\bullet$ *in* $P^{r,i}$ *is simply the other element.*

Lemma 4.5. *The total number of edges at any given time in a Theseus Clique is* $2n^2$.

Proof. We can proceed by induction. By construction E^0 has $2n^2$ edges since it is the union of two sets of size n^2; also, note that, by construction, each vertex has degree n.

We assume that E^l has $2n^2$ edges and every vertex has degree n. By construction, note that if an edge is in E^{l+1}, there is a corresponding edge in E^l and so we cannot have more than $2n^2$ edges. Now, we observe that $(a,b) \in E^{l+1}$ when $(a,b) \in E^l$ $a,b \notin P$, where P is the relevant swapping set. In these cases, we have clearly preserved the number of edges and the degree of each vertex. Now, we note that if $a \in P$, then for every edge that includes a in E^l, there is an edge that includes $\overline{a}$ in E^{l+1}; if there is an edge that includes $\overline{a}$ in E^l, then there is an edge that includes a in E^{l+1}. In other words, we have not changed the number of edges, nor the degree of any vertex. □

Lemma 4.6. *A Theseus Clique has two static connected components that are cliques at any given time step.*

Proof. We can proceed by induction. By construction, E^0 has two cliques. We assume that E^l has two cliques, say A, B. Let $a, b \in P$ be distinct. If a, b are in the same clique, then E^{l+1} trivially preserves cliques. If a, b are in different cliques, say $a \in A, b \in B$, then we observe that $(a,b) \notin E^l$ and so $(a,b) \notin E^{l+1}$. Next, we note that $(a,a') \notin E^{l+1}$ for any $a \in A$: because $(b,a) \notin E^l$ (by our inductive hypothesis), then $(\overline{a},a) \notin E^l$ because $\overline{a} = b$. Symmetrically, this must be true for b. Finally, we need only observe that by Lemma 4.5, since degree is preserved, then we must have maintained two cliques. □

Lemma 4.7. *For a Theseus Clique with total time steps* T *some positive multiple of* n^2, *then the number of edges between any two vertices* $v, v' \in V$ *is* $\frac{T}{2}$.

Proof. Every vertex has constant degree by Lemma 4.5. Because of Lemma 4.6, we know there are always two cliques. Now, we must show that for any pair of vertices, they are in the same clique exactly $\frac{T}{2}$ time steps. At the end of each round, we observe that our cliques are simply just U, V by construction. By construction, note that every vertex in U and every vertex in V exactly once per n^2 time steps. □

Theorem 4.8. *Given a Theseus Clique, the partition of vertices induced by the optimal solution to Expression 1 with* 2 *clusters has lower objective value than with* 1 *cluster.*

Proof. By construction, $\delta^t(u,v)$ is either 1 or ∞ if u, v are the in same clique at time t or not, respectively. The objective value with 1 cluster is therefore ∞, since there exists at least one vertex that is not in the same clique as the cluster center. There exists a finite objective value with 2 clusters: namely, at each time step, simply select a cluster center from each clique, which is well-defined by Lemma 4.6. □

4.4 Robust to Noise

Here, we use a similar setup as in Section 4.2, but we define a stochastic system, rather than a deterministic one. Theorem 4.11 shows that we still find the correct number of clusters in expectation. This result is somewhat weak, since we do not provide any analysis on the distribution of the number of clusters, nor do we provide a sensitivity analysis with respect to noise. That type of analysis is much more complicated, which we hope to provide in future work. However, our result provides at least some theoretical guarantees that STGkM is insensitive to noise, and we suspect that it is actually quite robust to noise based on our empirical results.

Definition 4.4 (Random Clique-cross-Clique)**.** *Let k be the "ground-truth" number of clusters and n be the number of vertices per cluster. We have a total of $N \triangleq k \times n$ vertices in V. For ease of indexing, we let $v_{i,j}$ be the j^{th} vertex in the i^{th} cluster, i.e. $i \in [k], j \in [n]$. We construct the following random dynamic graph: $P\left[(v_{i,j}, v_{i',j'}) \in E^t\right]$ is*

$$\begin{cases} 1 & i = i', j = j' \\ p & i = i', j \neq j' \\ p' & i \neq i', j = j' \\ 0 & i \neq i', j \neq j' \end{cases}$$

where $1 > p > p' > 0$. In other words, each edge at each time step is a Bernoulli random variable and we further require that each is independent. We will call such a random dynamic graph $\mathcal{G} = (V, E^t)$ a Random Clique-cross-Clique.

Lemma 4.9. *A Random Clique-cross-Clique is time-homogenous Markovian.*

Proof. For distinct time steps t, t' note that induced static graphs $G^t = (V, E^t), G^{t'} = (V, E^{t'})$ are independent by definition. Note that each edge is a Bernoulli random variable with probability that does not depend on time, so $G^t, G^{t'}$ are identically distributed. □

Lemma 4.10. *Given a Random Clique-cross-Clique with k clusters, p, p', vertices u, v, v' such that u, v are in the same cluster and u, v' are in distinct clusters, then $\mathbb{E}[\delta^t(u,v)] < \mathbb{E}[\delta^t(u,v')]$.*

Proof. In the trivial case where $u = v$, then $\mathbb{E}[\delta^t(u,v)] = 1$; since there is non-zero probability, namely $1 - p'$, that $(u, v') \notin E^t$, then $\mathbb{E}[\delta^t(u,v')] > 1$. We thus proceed by assuming that $u \neq v$.

First, let $X_{u,v}$ represent the first t such that $(u,v) \in E^t$[6] and let $X_{u,v'}$ be the analogous random variable for the edge u, v'. These are geometric random variables by construction and have well-defined expectations. Since $\delta^t(u,v)$ represents the shortest journey, we know that $\mathbb{E}[\delta^t(u,v)] \leq \mathbb{E}[X_{u,v}]$ because we can only reduce the possible shortest journey length by considering journeys that are not simply the edge between u, v because we almost surely have self-loops. Therefore, $\mathbb{E}[\delta^t(u,v)]$ is well-defined and, by similar logic, so is $\mathbb{E}[\delta^t(u,v')]$.

Next, we note that any journey between u, v is at least as likely to exist in the dynamic graph as any equivalent journey between u, v'. Because our dynamic graph is Markovian by Lemma 4.9, we can simply assume we are starting at time $t = 1$ without loss of generality. Let $j \triangleq (u, w_1, \ldots, w_{l-1}, v)$ be any journey of length l starting at time step t where no w_i is v or v'. Let $j' \triangleq (u, w_1, \ldots, w_{l-1}, v')$ be the related journey of length l. We observe that $P[(w_{l-1}, v) \in E^l] \geq P[(w_{l-1}, v') \in E^l]$ by construction.

[6] Formally, $X_{u,v} = \min\{t : (u,v) \in E^t\}$

Since any journey can be written in this form, any journey between u, v is at least as likely to exist in the dynamic graph as any journey between u, v'.[7]

Finally, we note that if any journey between u, v is at least as likely to exist as the equivalent journey between u, v', then $\mathbb{E}[\delta^t(u,v)] \leq \mathbb{E}[\delta^t(u,v')]$ by definition of a shortest journey. To wit, note that if a journey becomes more likely, then any case where the shortest journey is longer than it becomes less likely; in other words, if a journey j becomes more likely, then the probability that the shortest journey is longer than j simply cannot increase. Now, we need only observe that by definition of our random dynamic graphs that the edge (u, v) is strictly more likely than (u, v'), which means that we can conclude that $\mathbb{E}[\delta^t(u,v)]$ is strictly smaller than $\mathbb{E}[\delta^t(u,v')]$ since the shortest journey has length 1. More precisely, note that $\mathbb{E}[\delta^t(u,v)] = 1p + c(1-p), \mathbb{E}[\delta^t(u,v')] = 1p' + c(1-p')$ where c, c' represents the expected shortest journey length conditioned on the edge $(u,v), (u,v')$ not existing. Since we shows that $c \leq c'$ and by definition $p > p'$, then $\mathbb{E}[\delta^t(u,v)] < \mathbb{E}[\delta^t(u,v')]$. □

Theorem 4.11. *In expectation, given a Random Clique-cross-Clique with k clusters, p, p', the partition of vertices induced by the optimal solution to Expression 1 is exactly the k clusters with sufficient time.*

Proof. By Lemma 4.10, we know that $\mathbb{E}[\delta^t(u,v)] < \mathbb{E}[\delta^t(u,v')]$. By linearity of expectation, we can simply apply Theorem 4.3. □

5 Experimental Results

5.1 Choosing k

Perhaps the greatest challenge in using a k-means-based approach for clustering is determining the optimal number of clusters. With regard to classical k-means, two of the most common methods for choosing k are the Elbow Method [30], 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 and silhouette score [31, 32], which provides a measure of cluster cohesion versus separation.

In our previous work [1], we chose k using an approach similar to the Elbow Method but specialized for multi-cluster membership. Unlike classical k-means where increasing k results in points getting progressively closer to their centers, in mutli-membership STGkM, increasing k is likely to cause vertices to be assigned to progressively more clusters simultaneously, resulting in in a larger objective value. This allows us to search for a local minimum, as opposed to the "elbow," in our plot of k vs objective value.

In this work, because we choose to restrict vertices to single cluster membership, we can no longer follow the same procedure. Since the "elbow" can be very difficult to identify when using real data, we instead turn to the silhouette score [31, 32]. As a note, silhouette score ranges between 0 and 1, and a higher silhouette score is better. For each value of k, we calculate the average silhouette score over all time steps, and choose the number of clusters that results in the maximum average silhouette score. We repeat the experiments from [1] and show that they are consistent with our previous results.

[7] It is a tedious and technical matter to address the case where j includes v' and j' includes v. To do this, we first observe that if v appeared in j before the last vertex, then it would not be a proper journey of length l by definition. We then define a notion of journey equivalence where if j contains any instances of v', we swap these to v in j' and we show that we have not inadvertently increase the probability of j'. From here, we can conclude that our statement is still correct.

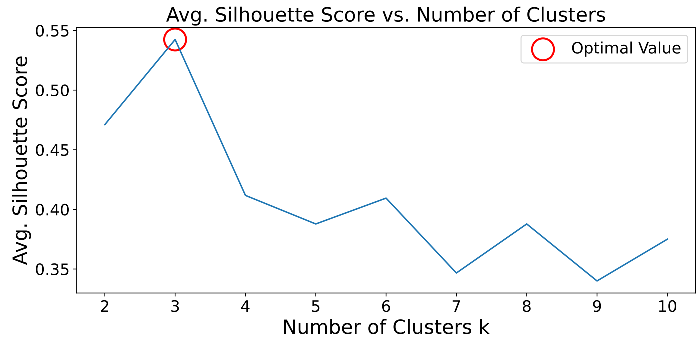

Fig 2. Average silhouette score versus number of clusters on a synthetic dataset.

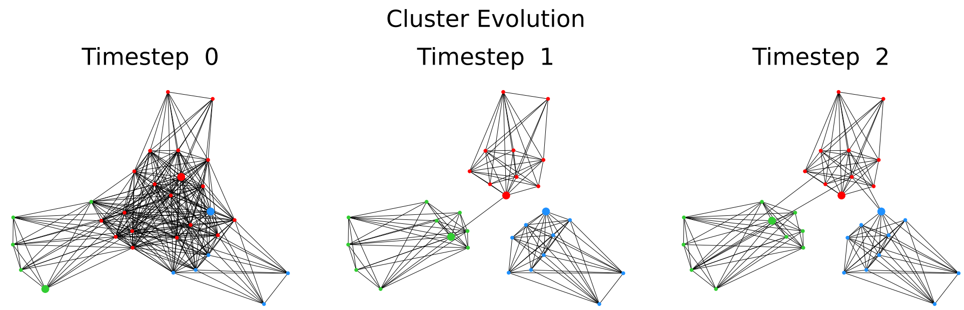

Fig 3. Three snapshots of a dynamic graph and the dynamic clustering as predicted by STGkM. Cluster centroids are identified by enlarged nodes.

5.2 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 Figure 2, and a snapshot of the evolution of detected clusters is shown in Figure 3. As expected, three communities persist throughout the duration of the simulation.

5.3 Detecting Political Parties

To demonstrate the utility of STGkM on a real-world dataset, we turn to the political sphere. 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 [33], 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 $k = 2$, corresponding to the two major US political parties,

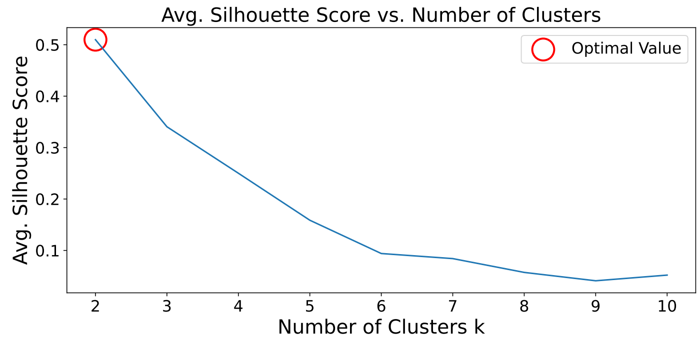

Fig 4. Average silhouette score versus number of clusters on the Roll Call dataset.

but in [1], when using multi-membership STGkM, our k selection process recommended $k = 3$. Beyond separating Democrats and Republicans into separate long-term clusters, we also identified an additional sub-cluster of three Democrats who very often vote "Present" together, as opposed to the majority Democratic party vote. Here, when using single-membership STGkM, we instead find $k = 2$ to be optimal, as shown in Figure 4. Interestingly, however, if we set $k = 3$, we are still able to recover the same sub-cluster of outlying Democrats, as in [1], up to stochasticity.

Figure 5 visualizes the similarity scores, as defined in Equation 3, between the cluster assignment histories for each pair of representatives when single-membership STGkM is run using $k = 2$ and $k = 3$. The rows and columns of the similarity matrices are ordered according to the discovered long-term communities. In the left figure, these clusters correspond to Republicans followed by Democrats, while in the right figure, the three outlying Democrats are moved 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. These figures agree with those generated in [1], using multi-membership STGkM.

5.3.1 Runtime

The difference in choice of optimal k between multi- and single-membership STGkM is intriguing. Perhaps restricting to single cluster membership leads to a loss of information, because we are forced to make immediate decisions about vertices that are equidistant from multiple centers. However, the computational speedup of restricting the search space cannot be ignored. When running multi- and single- membership STGkM on the Roll Call data 100 times each with $k = 2$, we find that the average runtime of multi-membership STGkM is about 1.36 seconds, while the average runtime of single-membership STGkM is .93 seconds, a speedup of 32%. We expect this computational advantage to become even more pronounced as our networks grow.

5.4 Detecting Journal Communities

For our next experiment, we explore communities based on citations between scientific journals. Using the Semantic Scholar API [34], we retrieve the 500 most cited papers for each year between 2000 and 2024 that are returned by the query "dynamic network." We then form a connectivity matrix based on whether there is a citation from a paper published in journal u to a paper published in journal v or vice versa in a given year. We also keep track of how many times such citations occur with a weight function defined as follows: for journals u, v with m total citations from papers in u to papers in

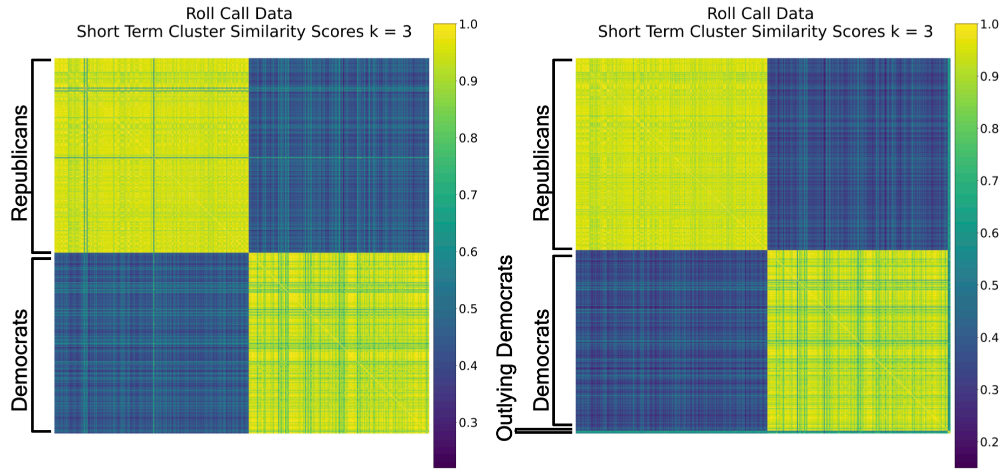

Fig 5. Short term clustering similarity between nodes in the Roll Call dataset using STGkM with $\gamma = 5$, $\lambda = 1$, and $k = 2$ on the left and $k = 3$ on the right.

v in year t, the s-journey between these two journals starting at time t will be scaled by $\frac{1}{m}$. Our final dataset tracks 199 journals over 24 years. The resulting connectivity matrix is sparse, with only 2.5% of entries containing nonzero values.

We set our maximum center drift and time connectivity to the default values, $\lambda = 1$ and $\gamma = 1$. When searching for an optimal number of clusters between 2 and 20, we repeatedly return a value close to the top of the range. For example, Figure 6 shows a situation where the optimal choice is $k = 18$, and we report results for this parameter choice. Figure 7 captures the short term cluster similarity scores between the cluster assignment histories for each pair of journals. The rows and columns of the similarity matrix are ordered according to the long-term clusters discovered by Phase 2 of STGkM. While we observe strong similarity within long-term clusters, we also observe a coarse, macro-structure in the similarity matrix. For instance, cluster 0, which contains 56% of journals covering a diverse range of topics, maintains above average similarity with most clusters, other than cluster 1, which has a distinct general biology focus. We also see strong similarities between pairs of long term-clusters, such as 1 and 2. Upon further investigation, these contain journals focused on the closely related topics of general biology and neuroscience, respectively. We look to the evolving contents of our clusters for further explanation.

Figure 8 tracks the four most common journals in a subset of our 18 dynamic clusters. We note no fewer than four clusters corresponding to biological topics, with separate clusters for chemistry, molecular biology, cells, and neuroscience. Other topic clusters cover communications, neural networks, operations research and wireless/mobile networks, and computer vision. The observation that journals covering similar topics are clustered together gives us confidence in STGkM's results.

One of the advantages of STGkM is that we can track a journal's cluster membership over time. Unsurprisingly, given the overlap in topic coverage between clusters, most journals' memberships change often. In fact, on average, a journal is a member of 5.51 clusters. Because the majority of journals interact with many clusters, instead of distinct boundaries, we maintain loose similarities between our long-term clusters, which helps explain the hierarchical structure in Figure 7. Digging further into which journals switch clusters most often, we find that `arXiv.org`, with it's general focus belongs to 10 different clusters at least once, while a highly specialized journal like `IEEE Transactions on Geoscience and Remote Sensing` never switches

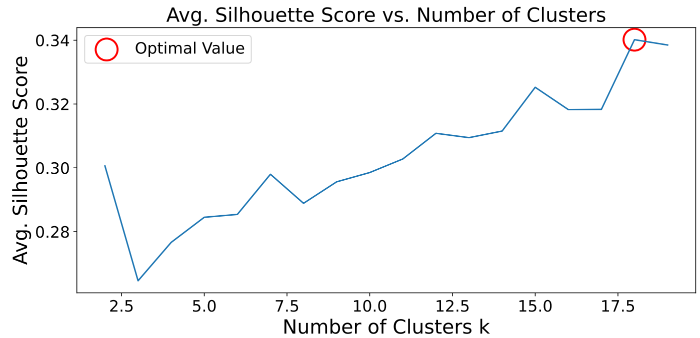

Fig 6. Average silhouette score versus number of clusters on the Semantic Scholar dataset.

membership. Perhaps, there is a loose relationship between the breadth of topics that a given journal covers and the stability of a journal's cluster assignment history.

5.5 Detecting Clusters of Subreddits

For our final experiment, we seek to find clusters of subreddits based on posts over a 2.5 year period between January 2014 and April 2017 [35]. We break the data into monthly windows and restrict to subreddits with at least 20 posts in the 2.5 year period. We define a connection between subreddit u and subreddit v if there is a hyperlink pointing to v present in the title of post u or vice versa. We use the same weight function as with the Semantic Scholar data, i.e. if subreddits link to one another m times within time period t, the s-journey between those two nodes starting at time t will be scaled by $\frac{1}{m}$. We provide two different sets of analysis based on whether the sentiment between the subreddits is positive or negative. The resulting positive sentiment connectivity matrix tracks the interactions of 1619 subreddits, while the negative sentiment connectivity matrix tracks the interactions of 115 subreddits, both over 41 months.

For both sets of analyses, we run STGkM with the default values $\lambda = 1$ and $\gamma = 1$. For the positive sentiment subreddits, our k selection process repeatedly recommends values of k towards the top end of our search range, and we report results for $k = 19$. For the negative sentiment subreddits, our k selection process consistently settles on $k = 2$. These processes are shown in Figure 9. One observation of note is that whereas the average silhouette scores for the negative sentiment subreddits are similar to those in previous experiments, the scores for positive sentiment subreddits are extremely low, suggesting that at most time steps, nodes are very difficult to separate into clusters. This hypothesis is strengthened by Figure 10, which shows the short term cluster similarity scores for both the positive and negative sentiment Reddit data, ordered by long-term cluster membership. Almost all positive sentiment subreddits belong to one long-term cluster, while negative sentiment subreddits are contained to two distinct long-term clusters. Since STGkM tracks both short- and long-term interactions between nodes, we turn to the content of our dynamic clusters for an explanation.

The first row of Figure 11 shows the four most common positive sentiment subreddits over time in a selection of four clusters. The first cluster pertains to negative Donald Trump content, the second to operating systems and coding, the third to conservative political content, and the fourth to sports. The second row of Figure 11 shows the ten most common subreddits over time in our two dynamic negative sentiment clusters. The contents of the first cluster are primarily political, as opposed to the contents of the second, even though there is some overlap. For instance, "btc" (bitcoin) is one of the most popular subreddits in both. These results suggest that in

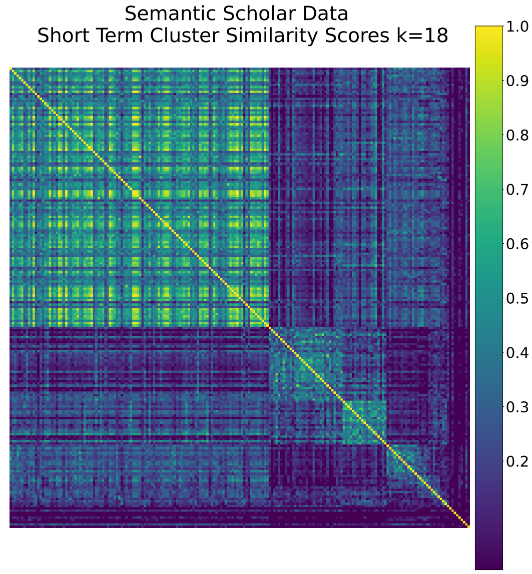

Fig 7. Short term clustering similarity between nodes in the Semantic Scholar dataset using STGkM with $\gamma = 1$, $\lambda = 1$, and $k = 18$.

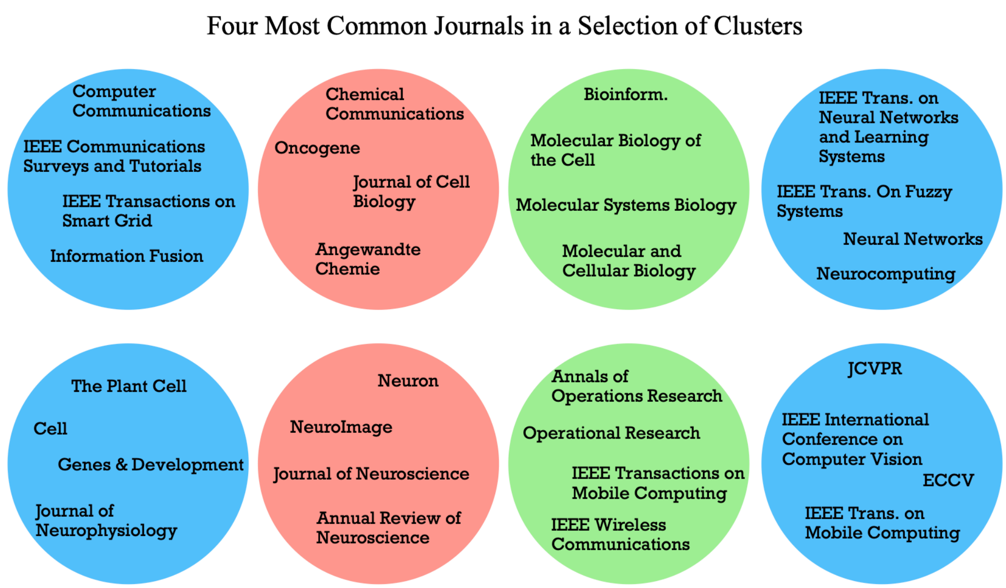

Fig 8. Four most common journals for a selection of clusters from the Semantic Scholar dataset using STGkM with $\gamma = 1$, $\lambda = 1$, and $k = 18$. The most common journals in each cluster tend to cover similar topics.

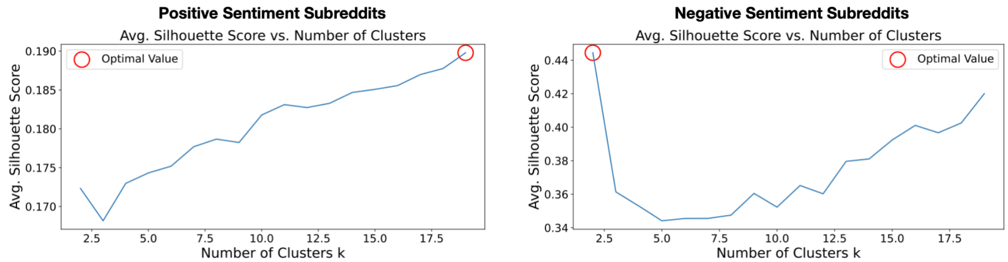

Fig 9. Average silhouette score verses number of clusters on the Reddit dataset for subreddits with positive sentiment on the left and negative sentiment on the right.

Fig 10. Short term clustering similarity between nodes in the Reddit dataset using STGkM with $\lambda = 1$, $\gamma = 1$, and $k = 19$ on positive sentiment data on the left and $\lambda = 1$, $\gamma = 1$, and $k = 2$ on negative sentiment data on the right.

both experiments, STGkM appropriately separates subreddits of different topics. The behaviour of our long-term clusters in Figure 10 can be explained by how often subreddits switch clusters. n average, positive sentiment subreddits belong to 5.88 clusters. In contrast, the cluster membership of negative sentiment subreddits is much more stable. The idea that negative sentiment is much more polarizing than positive sentiment is well studied [3]. Therefore, our observation that posts containing negative sentiment towards similar topics have much more stable clustering histories than posts containing positive content is unsurprising.

6 Conclusion

In [1], we introduced spatiotemporal graph k-means (STGkM) for community detection by vertex clustering on dynamic graphs. The 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. In this work, we improve the runtime and thereby the feasibility of running STGkM on larger networks by embracing a new relaxation and carry out experiments on one synthetic and three

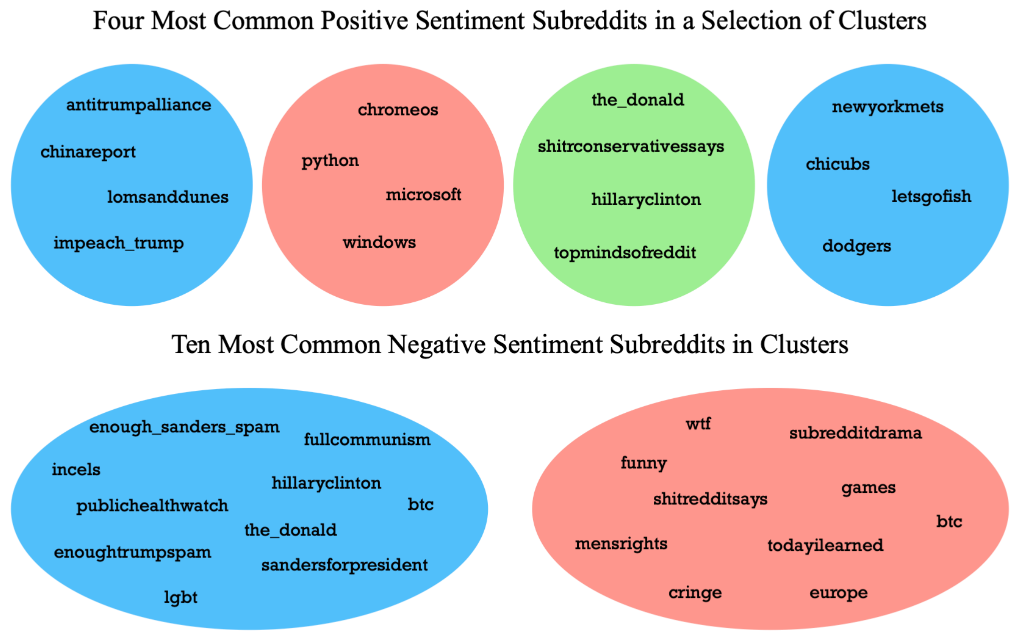

Fig 11. Top row: Four most common subreddits for a selection of clusters from the positive sentiment Reddit data using STGkM with $\gamma = 1$, $\lambda = 1$, and $k = 19$. Bottom row: Ten most common subreddits for the two dynamic clusters from the negative sentiment Reddit data using STGkM with $\gamma = 1$, $\lambda = 1$, and $k = 2$.

real-world datasets to empirically validate performance.

We also extend our theoretical guarantees and explain clustering behavior under certain conditions. While in our conference paper, we show that the partition induced by STGkM is identical to connected components in certain cases, in this work, we distinguish the cases where we can find sensible clusters that dynamic connected components cannot. We also provide a result that demonstrates why dynamic clustering may find better partitions than a static clustering technique; in particular, static clustering methods that aggregate or "average" behavior and then cluster the resulting graph cannot capture important dynamics. Finally, we give an initial result demonstrating that STGkM is robust to noise.

In all of our experiments, we find that STGkM is able to detect informative clusters and make interesting conclusions about trends in the datasets, such as detecting outlying political activity, the extent of similarity between subtopics of scientific journals, and supporting evidence of polarization in negative versus positive sentiment social media posts. These mathematical results align with the existing literature in each task's respective area of study, which provides further confidence that STGkM is finding "useful" clusters for real applications. These conclusions would not be possible without STGkM's ability to break down the multi-scale relationships between graph nodes.

One direction for further study is to extend STGkM to the online case. Because clustering is carried out at each time step independently, relying only on the centers at the previous time step to seed the current choice, it is feasible to transform STGkM to an online algorithm. The main challenges will be in updating s-journeys and deciding how many time steps of a dynamic graph to maintain and collect between cluster updates, but we leave this for future work. Inspired by the results of our experiments on the Semantic Scholar and Reddit data, another potential extension is a hierarchical version of STGkM. In the Semantic Scholar experiments, we already saw promising hierarchical structure within the short term cluster similarity matrix. On the other hand, in the Reddit experiments, we observed one massive long-term cluster, but dynamic short-term clusters with distinct topic focuses. We hypothesize that by intelligently applying STGkM to smaller and smaller subsets of data, we could more clearly extract the evident hierarchical relationships in the Semantic Scholar data and perhaps further break down the mega-cluster in the Reddit data.

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 also 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. Finally, we would like to provide much more precise results in the stochastic setting with more specific bounds on STGkM's output based on noise. We will also explore online extensions of STGkM, where we explore dynamic graphs in real-time. We intend to leverage STGkM in a variety of applications and we hope that it becomes an interesting tool in the study and analysis of various network-based data for data practitioners.

Acknowledgments

We would like to thank Albert Azout and Liam Shalon at Level Ventures and J. Nathan Kutz at the University of Washington for their support.

References

1. Dabke DV, Dorabiala O. A Novel Method for Vertex Clustering in Dynamic Networks. In: Complex Networks & Their Applications XII. Springer; 2023. p. TBA.

2. Dabke DV, Dorabiala O. Spatiotemporal Graph k-means. In: Proceedings of the Communities in Networks ComNets @ NetSci 2023; 2023. p. none available.

3. Baumeister RF, Bratslavsky E, Finkenauer C, Vohs KD. Bad is stronger than good. Review of general psychology. 2001;5(4):323–370.

4. Hylton A, Short R, Cleveland J, Freides O, Memon Z, Cardona R, et al. A Survey of Mathematical Structures for Lunar Networks. In: 2022 IEEE Aerospace Conference (AERO); 2022. p. 1–17.

5. Cleveland J, Hylton A, Short R, Mallery B, Green R, Curry J, et al. Introducing Tropical Geometric Approaches to Delay Tolerant Networking Optimization. In: 2022 IEEE Aerospace Conference (AERO); 2022. p. 1–11.

6. Dabke DV. On Systems of Dynamic Graphs: Theory and Applications [PhD thesis]. Princeton University. Fine Hall, 2nd Floor, Princeton, NJ, USA; 2023. Available from: `https://dataspace.princeton.edu/handle/88435/dsp010z709070k`.

7. Dabke DV, Chazelle B. Extracting Semantic Information from Dynamic Graphs of Geometric Data. In: Complex Networks & Their Applications X. Springer; 2021. p. 474–485.

8. Dabke DV, Arroyo EE. Rumors with Personality: A Differential and Agent-Based Model of Information Spread through Networks. SIAM Undergraduate Research Online. 2016;9:453–467. doi:10.1137/16S015103.

9. Dabke DV, Karntikoon K, Aluru C, Singh M, Chazelle B. Network-Augmented Compartmental Models to Track Asymptomatic Disease Spread. Bioinformatics Advances. 2023;doi:10.1093/bioadv/vbad082.

10. Fortunato S. Community detection in graphs. Physics reports. 2010;486(3-5):75–174.

11. Görke R, Maillard P, Schumm A, Staudt C, Wagner D. Dynamic graph clustering combining modularity and smoothness. Journal of Experimental Algorithmics (JEA). 2013;18:1–1.

12. Rossetti G, Cazabet R. Community discovery in dynamic networks: a survey. ACM computing surveys (CSUR). 2018;51(2):1–37.

13. Chakrabarti D, Kumar R, Tomkins A. Evolutionary clustering. In: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining; 2006. p. 554–560.

14. Lin YR, Chi Y, Zhu S, Sundaram H, Tseng BL. Facetnet: a framework for analyzing communities and their evolutions in dynamic networks. In: Proceedings of the 17th international conference on World Wide Web; 2008. p. 685–694.

15. Chi Y, Song X, Zhou D, Hino K, Tseng BL. Evolutionary spectral clustering by incorporating temporal smoothness. In: Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining; 2007. p. 153–162.

16. Ruan B, Gan J, Wu H, Wirth A. Dynamic structural clustering on graphs. In: Proceedings of the 2021 International Conference on Management of Data; 2021. p. 1491–1503.

17. 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; 2021. p. 4608–4616.

18. DiTursi DJ, Ghosh G, Bogdanov P. Local Community Detection in Dynamic Networks. In: 2017 IEEE International Conference on Data Mining (ICDM); 2017. p. 847–852.

19. Dorabiala O, Webster J, Kutz N, Aravkin A. Spatiotemporal k-means. arXiv preprint arXiv:221105337. 2022;.

20. Casteigts A, Flocchini P, Quattrociocchi W, Santoro N. Time-varying graphs and dynamic networks. International Journal of Parallel, Emergent and Distributed Systems. 2012;27(5):387–408.

21. Becker R, Casteigts A, Crescenzi P, Kodric B, Renken M, Raskin M, et al. Giant components in random temporal graphs. arXiv preprint arXiv:220514888. 2022;.

22. 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 '15. New York, NY, USA: Association for Computing Machinery; 2015. p. 475–489. Available from: `https://doi.org/10.1145/2723372.2737791`.

23. Lerman K, Ghosh R, Kang JH. Centrality Metric for Dynamic Networks. In: Proceedings of the Eighth Workshop on Mining and Learning with Graphs. MLG '10. New York, NY, USA: Association for Computing Machinery; 2010. p. 70—-77. Available from: `https://doi.org/10.1145/1830252.1830262`.

24. Bergamini E, Meyerhenke H. Approximating Betweenness Centrality in Fully Dynamic Networks. Internet Mathematics. 2016;12(5):281–314. doi:10.1080/15427951.2016.1177802.

25. Yen CC, Yeh MY, Chen MS. An Efficient Approach to Updating Closeness Centrality and Average Path Length in Dynamic Networks. In: 2013 IEEE 13th International Conference on Data Mining; 2013. p. 867–876.

26. Habiba, Tantipathananandh C, Berger-Wolf TY. Betweenness Centrality Measure in Dynamic Networks. DIMACS; 2007. 19.

27. Latapy M, Viard T, Magnien C. Stream graphs and link streams for the modeling of interactions over time. Social Network Analysis and Mining. 2018;8(1):61. doi:10.1007/s13278-018-0537-7.

28. Megiddo N, Supowit KJ. On the complexity of some common geometric location problems. SIAM journal on computing. 1984;13(1):182–196.

29. Schubert E, Rousseeuw PJ. Fast and eager k-medoids clustering: O (k) runtime improvement of the PAM, CLARA, and CLARANS algorithms. Information Systems. 2021;101:101804.

30. Kodinariya TM, Makwana PR, et al. Review on determining number of Cluster in K-Means Clustering. International Journal. 2013;1(6):90–95.

31. Rousseeuw PJ. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. Journal of computational and applied mathematics. 1987;20:53–65.

32. Lenssen L, Schubert E. Medoid Silhouette clustering with automatic cluster number selection. Information Systems. 2024;120:102290.

33. 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. Wiley Online Library; 2011. p. 1061–1070.

34. Kinney R, Anastasiades C, Authur R, Beltagy I, Bragg J, Buraczynski A, et al. The semantic scholar open data platform. arXiv preprint arXiv:230110140. 2023;.

35. Kumar S, Hamilton WL, Leskovec J, Jurafsky D. Community interaction and conflict on the web. In: Proceedings of the 2018 World Wide Web Conference on World Wide Web. International World Wide Web Conferences Steering Committee; 2018. p. 933–943.