

Local Community Detection in Dynamic Networks

Daniel J. DiTursi*[†]

*Department of Computer Science
State University of New York at Albany
Albany, NY 12222

Gaurav Ghosh*

[†]Department of Computer Science
Siena College
Loudonville, NY 12211

Petko Bogdanov*

Abstract—Given a time-evolving network, how can we detect communities over periods of high internal and low external interactions? To address this question we generalize traditional local community detection in graphs to the setting of dynamic networks. Adopting existing static-network approaches in an “aggregated” graph of all temporal interactions is not appropriate for the problem as dynamic communities may be short-lived and thus lost when mixing interactions over long periods. Hence, dynamic community mining requires the detection of both the community nodes and an optimal time interval in which they are actively interacting.

We propose a filter-and-verify framework for dynamic community detection. To scale to long intervals of graph evolution, we employ novel spectral bounds for dynamic community conductance and employ them to filter suboptimal periods in near-linear time. We also design a time-and-graph-aware locality sensitive hashing family to effectively spot promising community cores. Our method PHASR discovers communities of consistently higher quality (2 to 67 times better) than those of baselines. At the same time, our bounds allow for pruning between 55% and 95% of the search space, resulting in significant savings in running time compared to exhaustive alternatives for even modest time intervals of graph evolution.

I. INTRODUCTION

Given a large network with entities interacting at different times, how can we detect communities of intense internal and limited external interactions over a period? Temporal network interaction data abounds, hence, answering the question above can inform decisions in a wide range of settings. A set of computers that do not typically interact extensively suddenly exhibits a spike in network traffic—this burst could be the activation of a botnet and warrants special attention from network administrators [15]. Similarly, prior to a major project deadline, members of a team may communicate more and exclusively among each other compared to other times [19]. Knowledge of such periods and the involved parties can lead to better communication systems by accordingly ranking temporally and contextually important messages.

In the example of Fig. 1 nodes $\{2, 3, 4\}$ induce a strong community at times t_2 - t_3 due to multiple high-weight (thick line) internal and weaker external interactions. Intuitively, including more nodes or extending this community in time can only make it less exclusive. Detection of such dynamic communities can be viewed as a generalization of local community detection [3, 2], where locality is enforced in both (i) the graph domain: local as opposed to complete partitioning; and (ii) the time domain: communities exhibit bursty internal interactions in a contiguous time interval. For example, in a log of cellular tower interactions in the city of Milan, we detect just such

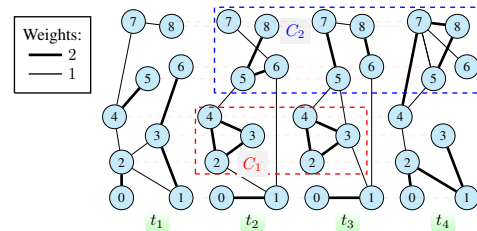


Fig. 1. Interactions in a small graph over 4 time steps. Thicker edges designate stronger interactions. Two temporal communities C_1 across t_2 - t_3 and C_2 across t_2 - t_4 are designated in dashed boxes.

a bursty temporal community near a major highway during hours corresponding to the morning commute. (See Fig. 2.)

Unlike evolutionary clustering [21, 18, 6], whose goal is to partition all vertices at every timestamp, our goal is to identify the most cohesive communities and their interval of activity without clustering all nodes in time. Hence, our problem is more similar to local community detection [3, 13, 29] than partitioning. In addition, evolutionary clustering methods are often concerned with the long-term group membership evolution: how partitions appear, grow, shrink and disappear [6]. Instead, we focus on interaction bursts among a temporally stable group of nodes.

While local dynamic community detection has practical applications, it also presents non-trivial challenges. Even in static graphs, many local community measures involving cuts are NP-hard to optimize, including conductance [30], modularity [8], ratio cut [31], and normalized cut [27]. Furthermore, to detect the active period of a dynamic community one needs to consider a quadratic number of possible intervals. Aggregating all interactions and resorting to static community detection approaches [3, 13] may occlude dynamic communities due to mixing interactions from different periods. Alternatively, consideration of individual timestamps in isolation may fragment the community in time.

We propose a *Prune, HASH and Refine (PHASR)* approach for the problem of *temporal community detection*. We *prune* infeasible time periods based on novel spectral lower bounds for the graph conductance tailored to the dynamic graph setting. We show that pruning all $O(T^2)$ possible intervals can be performed in time $O(T \log T)$ due to an interval grouping scheme exploiting the similarity of overlapping time intervals. In order to efficiently *spot* candidate community nodes in non-pruned time intervals, we design a *time-and-graph-aware* locality sensitive hashing scheme to group similar temporal neighborhoods of community nodes in linear time. Our hash-

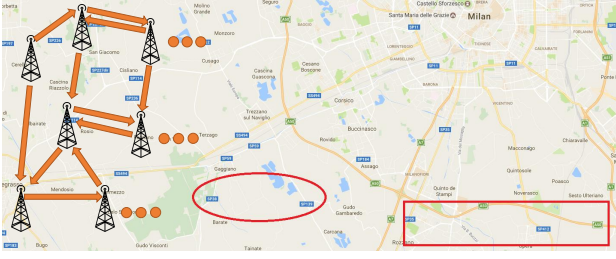


Fig. 2. Map of Milan and surroundings. The marked areas contain the cell towers of the top discovered communities in the *Call* data set. These communities exist in the early morning hours and likely correspond to morning commute along Milan’s beltway.

ing scheme can be configured to maximize the probability of spotting communities of a target duration informed by the pruning step. In the *refinement* step, we expand communities rooted in candidate nodes in time. Our contributions in this work are as follows:

- We propose novel spectral bounds for temporal community conductance and an efficient scheme to compute them using $O(T \log T)$ eigenvalue computations. Our bounds enable pruning of more than 95% of the time intervals in synthetic and more than 50% in real-world instances; and are trivial to parallelize.
- We propose a joint time-and-graph, locality-sensitive family of functions and employ them in an effective scheme for spotting temporal community seeds in linear time. Our LSH scheme enables the discovery of 2 to 67 times better communities compared to those discovered by baselines.
- PHASR scales to synthetic and real-world networks of sizes that render exhaustive alternatives infeasible. This dominating performance is enabled by effective pruning and high true positive rate of candidates produced by our hashing scheme.

II. RELATED WORK

Static local communities: Our work is different from static community detection [32, 3, 23, 13, 20] in that we consider a dynamic setting. We compare to temporal generalization of the method in [3] since it similarly focuses on the subgraph conductance of a community and the method in [23] as it employs hashing for static network communities. Our experiments demonstrate that naive extensions of the above methods do not scale well with time.

Dense subgraph detection has been recently considered for unweighted interactions in time [25, 14, 24]. Such methods allow for analysis at the maximal temporal resolution in which only fragments of the community may be available at an instance and thus require parameters that predefine the total span of a community [25] or some notion of persistence (e.g. time-to-live interval) for interaction edges [14, 24]. In addition, these works consider the density within a community, but not its separation from the rest of the network, i.e. its cut. In contrast, in our setting we focus on communities that are well-separated from the rest of the network, we do not require predefined persistence/span, and we allow for weights of edges modelling varying strength of interactions in time.

Persistent subgraphs in time have also been considered [22, 1]. Similar to this work, the methods in this category require that the subgraph of interest persists as either a conserved topology [1] or as a stable level of intra-community interactions. These objectives are different and even complementary to ours in that they do not consider how well-separated the community is from the rest of the network. The persistence requirements can be considered in conjunction with conductance to further rank candidates of interest.

High-weight temporal subgraph detection is another relevant setting with the goal of detecting connected subgraphs which optimize a function of their node or edge weights in time have also been considered [22, 1, 7]. Similar to this work, the methods in this category require that the subgraph of interest persists as either a conserved topology [1] or as a stable level of intra-community interactions. These objectives are different from ours in that they do not consider how well-separated the community is from the rest of the network; the focus is only on high internal weights. The persistence requirements could be considered in conjunction with conductance to further rank candidates of interest.

Evolutionary clustering for dynamic networks is another related and very active area of research [21, 18, 6, 29]. The goal in evolutionary clustering is to track the changes in the global network partitions over time, where partitions are allowed to vary from one time slice to the next by incorporating temporal smoothness of partition membership. The general problem setting, however, differs from ours as the goal is to partition all vertices at every timestamp as opposed to identifying the best local communities and their interval of activity. Closest to our goal from this group is the local community method by Takafolli et al. [29] which extends local communities of good modularity from one time-step to the next. In comparison, our approach considers the full timeline as opposed to only consecutive time steps, and as a result consistently finds lower-conductance communities than that of Takafolli et al. [29].

III. PROBLEM DEFINITION

Our goal is to find communities of stable membership over a period of time during which members interact mostly among each other as opposed to with the rest of the network. To model this intuition we propose the *temporal conductance* measure, a natural extension to graph conductance which is commonly adopted for local communities in static graphs [30, 3]. Our problem can then be cast as detecting the subgraph and interval of smallest temporal conductance, formalized next.

Let $G(V, E, W)$ be an undirected edge-weighted *temporal graph*, where V is the set of vertices, $E \subseteq V \times V$ is the set of edges and W is a family of weight functions $W : E \times T \rightarrow \mathbb{R}^+$ mapping edges to real values across a discrete *timeline* $\mathcal{T} = \{0, 1, \dots, |T| - 1\}$ of graph evolution. We will use $w(u, v, t)$ to denote the weight of an edge (u, v) at time t and $w(u, v, t, t') = \sum_{i=t}^{t'} w(u, v, i)$ to denote the aggregate (temporal) weight on the same edge in the interval $[t, t']$. The temporal volume of a node u is defined as

$vol(u, t, t') = \sum_{(u,v) \in E} w(u, v, t, t')$. A *temporal community* (C, t, t') is a connected subgraph of G induced by nodes $C \subseteq V$ and weighted by $w(u, v, t, t'), \forall (u, v) \in E \cap (C \times C)$. The *temporal conductance*, of a community (C, t, t') , a generalization of the classic conductance [30], is defined as:

$$\phi(C, t, t') = \eta(t, t') \frac{cut(C, t, t')}{\min(vol(C, t, t'), vol(\bar{C}, t, t'))},$$

where $\bar{C} = V \setminus C$; $cut(C, t, t') = \sum_{u \in C, v \in \bar{C}} w(u, v, t, t')$ is the temporal cut of C ; and $\eta(t, t')$ is a temporal normalization factor. The smaller the conductance, the more cohesive the community.

If $\eta(t, t')$ is a constant, the temporal conductance reduces to the regular graph conductance of C on an aggregated network in the interval $[t, t']$. However, without normalization the conductance will favor small communities in single timestamps, thus fragmenting a natural community in time. Hence, we consider a temporal normalization $\eta(t, t') = (t' - t)^{-\alpha}$, where α controls the importance of community time extent. Our methods can trivially accommodate different forms of the normalization function, e.g. exponential time decay similar to that used in streaming settings [26, 33].

To demonstrate the effect of normalization, consider C_1 and C_2 in Fig. 1. When $\alpha = 0$ (i.e. no normalization), the conductance of C_1 is $\phi(C_1, 2, 3) = 5/29 = 0.17$, while $\phi(C_2, 2, 4) = 8/39 = 0.21$ (weights considered). Upon increasing the normalization (say $\alpha = 1$), and hence the preference for longer-lasting communities, the temporal conductance of C_2 becomes lower than that of C_1 .

Problem 1. [Lowest temporal conductance community]

Given a dynamic network $G(V, E, W)$, find the community: $(C_o, t_o, t'_o) = \arg \min_{C \in V, 0 \leq t \leq t' \leq T} \phi(C, t, t')$.

The *lowest conductance* problem in a static graph is known to be NP-hard [30] and since a dynamic graph of a single timestamp $T = 1$ is equivalent to the static case, our problem of *temporal conductance minimization* is also NP-hard. Hence, our focus is on (i) scalable processing of dynamic graphs over long timelines; and (ii) effective and efficient detection of community seeds in the graph and time which existing approximate solutions for the static case require as input [3, 2].

IV. METHODS

Since our problem is NP-hard, exhaustive approaches would not scale to large real-world dynamic networks. We experimentally demonstrate that naïve heuristics are infeasible in all but trivially-small instances. Thus, our overall method enables scalability (i) with the graph size by identifying and refining candidate seed nodes in time that are likely to participate in low-conductance communities (*Seed selection IV-B*); and (ii) with the length of the timeline by pruning infeasible periods in time with guarantees (*Pruning Sec. IV-C*). Our final approach is presented in Sec. IV-D.

A. Preliminaries

Before we present our solution, we review preliminaries related to *locality sensitive hashing (LSH)* [16] and *local community detection* [3, 4]. Both concepts have been employed

for static networks and are, thus, a natural starting point for naïve baselines.

LSH and neighborhood similarity. Indyk et al. [16] proposed LSH for approximate nearest neighbor (NN) search. A family of functions is (d_1, d_2, p_1, p_2) -sensitive w.r.t. a distance measure d if for every function f in the family, $d_2 \leq d(x, y) \leq d_1 \Rightarrow p_2 \geq P[f(x) = f(y)] \geq p_1$. The family of *minhash* functions for sets was shown to be locality-sensitive w.r.t. the Jaccard distance (defined as 1–Jaccard Similarity) [9, 16]. The concept of LSH for node neighborhoods was employed for fast community detection in static networks by Macropol et al. [23]. The intuition is that nodes in dense communities tend to have similar neighborhoods and thus they will collide when hashed using an LS family. We extend this intuition to locality in time by considering similarity in both time and graph space. In addition, the community strength in time is dependent not only on the existence of edges but also the level of interaction, which we model as weights in time $w(u, v, t)$. In order to incorporate weights we adopt a recent approach by Ioffe et al. [17] for LSH of weighted sets using a weighted Jaccard similarity, defined as follows for neighborhoods in our setting:

$$J_W(N_i, N_j) = \frac{\sum_{v \in N_i \cup N_j} \min(w(v, i), w(v, j))}{\sum_{v \in N_i \cup N_j} \max(w(v, i), w(v, j))},$$

where N_i and N_j are the neighborhoods of nodes i and j in a given time period (time indices of the weight omitted for simplicity).

Local communities in static graphs. Recent approaches for low-conductance local community detection in static networks rely on graph diffusion [3, 4]. The goal is to obtain a single partition around a predefined seed node by a local computation that involves a small fraction of the graph around the seed and that closely approximates the best conductance involving the seed. In our proposed approach, we first “spot” seed nodes in time based on high temporal neighborhood similarity and expand to a community similar to the spectral sweep method by Andersen et al. [3].

B. Temporal Neighborhood LSH to spot seeds

A low-conductance temporal community consists of nodes that mostly interact with each other over a contiguous time interval when the community is active. A first step in our approach is to find seed nodes within the community in the corresponding time frame that can then be used to expand to strong communities. Our solution for seed selection is based on the observation that weighted node neighborhoods within the community tend to be similar. We exploit this observation in order to obtain seeds for promising regions in time. Specifically, we propose a scalable similarity search method based on hashing of node neighborhoods in time. We show that our scheme is locality-sensitive in both time and graph space. Its parameters can be optimized to target a pre-specified duration in time—a property that we exploit in conjunction with temporal pruning of feasible intervals in order to reduce the computational and memory footprint of our approach.

Given a dynamic graph $G(V, E, W)$, the weighted temporal neighborhood N_u^t of node u is the weighted set of its neighbors (including u): $N_u^t = \{(v : w(u, v, t)) | (u, v) \in E\} \cup (u : vol(u, t))$. We adopt the weighted Jaccard similarity $J_W(N_u^t, N_v^t)$ and the weighted minhash function for $\psi(\cdot)$ ensuring that

$$P[\psi(N_u^t) = \psi(N_v^t)] = J_W(N_u^t, N_v^t).$$

We hash neighborhoods using r independent minhash functions to create a graph signature $S_G^r(N_u^t)$ for a given neighborhood N_u^t .

If we compare weighted neighborhoods, disregarding the time at which they were observed, we may produce collisions of high-similarity neighborhoods that may be potentially distant in time. Hence, a straightforward adoption of LSH for weighted sets will not be locality sensitive with respect to time. Instead, we need to associate highly similar neighborhoods that are also close in time. The main intuition behind our temporal locality sensitive hashing function is: *close time instants are likely to belong to the same interval if the timeline is partitioned into random segments*.

Let $p^k = \{p_1 < p_2 \dots < p_k\}$ be a k -partitioning of the timeline using k pivot time points selected uniformly at random in $[0, T]$. We define a hash function $\tau^k(\cdot)$ based on the partitioning p that maps a given time point t to the index of the earliest pivot $p_i \in p$ whose time exceeds t : $\tau^k(t) = \{\min(i) | p_i \geq t, p_i \in p^k\}$.

Theorem 1. [Temporal locality] *The family of temporal hash functions τ^k is $(\Delta_1, \Delta_2, (1 - \frac{\Delta_1}{T})^k, (1 - \frac{\Delta_2}{T})^k)$ -sensitive family for the distance in time Δ defined as the delay between two timepoints.*

Proof. Available in the Appendix. \square

Beyond being locality sensitive in time, our pivot-based hashing family τ^k can be configured to target specific community lengths. While we do not know the duration of good communities in the data a priori, as we will show in the following section, we can prune intervals in time that cannot include the best communities with guarantees. Hence, we need to be able to focus on matching neighborhoods at time resolutions that are viable in order to reduce the memory and running time footprint of our solution. To enable this, we need to answer the following question: *What is the optimal number of pivots k to detect communities of a given duration?* Assuming that a target community duration is Δ^* , we need to choose k such that similar temporal neighborhoods within that period have a high chance of collision. A perfect partitioning p of the timeline would produce a single segment that isolates the target period of length Δ^* . This requires two pivots to “bracket” the period and all other pivots to fall outside of it.

Theorem 2. [Optimal number of pivots] *The number of pivots k^* that maximizes the probability of a perfect partition of a period of length Δ^* is $k^* \approx \lfloor \frac{2T}{\Delta^*} \rfloor$.*

Proof. Available in the Appendix. \square

To detect similar neighborhoods in time we combine an r -sized graph signature $S_G^r(N_u^t)$ with a temporal signature $S_T^k(N_u^t) = \tau^k(t)$ using an *AND* predicate to obtain a unified

temporal neighborhood signature $S^{r,k}(N_u^t)$ that is a locality sensitive family in both the time and graph domains as a direct consequence of the locality $S_G^r(N_u^t)$ and $S_T^k(N_u^t)$.

Corollary 3. *Let $S^{r,k}$ be a temporal neighborhood hash function with r minhashes and k partitions. Then: $P[S^{r,k}(N_u^t) = S^{r,k}(N_u^{t'})] = J_W(N_u^t, N_u^{t'})^r (1 - \frac{|t-t'|}{T})^k$.*

Since our composite hashing family is locality sensitive in both time and the graph, we can construct signatures that amplify its locality sensitivity. For example, if we combine l independent hash signatures $S^{r,k}$ using an *OR* predicate, i.e. require that there is a match in at least one hash value for a collision, the resulting collision probability will be $1 - [1 - p_{r,k}]^l$, where $p_{r,k} = P[S^{r,k}(N_u^t) = S^{r,k}(N_u^{t'})]$. Similarly, an *AND* predicate composition will result in $p_{r,k}^l$ probability of collision. Using cascades of such composition we can “shape” the selectivity of our LSH scheme and thus control rate of FP and FN collisions at the cost of increased memory and computational overhead.

A neighborhood signature requires $\log(k|V|^r)$ bits of storage, since a single weighted minhash value is a vertex index and a temporal hash value is the position of the first pivot index exceeding the timestamp of the hashed neighborhood. For a fixed temporal resolution and a composition of b independent signatures (OR/AND predicate compositions), the overall memory footprint of our hashing approach will be $bT|V| \log(k|V|^r)$ bits. The above analysis is pessimistic as it assumes no collisions and thus storing the signatures for all existing collision bins. Nevertheless, large and long-evolving instances may be impractical to exhaustively hash and process. Our filtering approach discussed next addresses this challenge and allows us to significantly reduce the memory and computational footprint.

C. Spectral bounds for pruning time intervals

The strongest temporal communities exhibit lower conductance than other communities in the network. Also, in real-world graphs many time intervals contain no promising communities—i.e. no project deadline or spike in network traffic. Our goal is to eliminate from consideration such periods of low community activity which cannot coincide with the best temporal community. In what follows, we develop lower bounds on the temporal conductance of any subgraph in a given time window. We employ our bounds in combination with a solution estimate to deterministically prune irrelevant intervals in time. Such pruning can significantly improve the running time of our LSH-based approach as we can target only promising neighborhoods in time by adjusting the time scale (i.e. number of pivots k) for temporal hashing.

Let $G^{[t,t']}$ be the aggregate graph of $G(V, E, W)$ over time interval $[t, t']$ with aggregate edge weights $w(u, v, t, t')$. The temporal *graph conductance* in an aggregate weighted graph is defined as the minimum temporal conductance over all subsets $C \in V$: $\phi(G^{[t,t']}) = \min_{C \in V} \phi(S, t, t')$. Let A be the adjacency matrix of a weighted graph $G^{[t,t']}$ with elements $A_{u,v} = w(u, v, t, t')$ and D be the diagonal “degree”

matrix with elements $D_{u,u} = \text{vol}(u, t, t')$ and 0 in all off-diagonal elements. The matrix $\mathcal{L} = D - A$ is the unnormalized graph Laplacian, while the matrix $\mathcal{N} = D^{-1/2} \mathcal{L} D^{-1/2}$ is the symmetric normalized graph Laplacian [28]. The Laplacian matrices have many advantageous properties and have been employed in spectral graph partitioning [28, 11]. The eigenvalues $0 = \lambda_1 \leq \lambda_2 \leq \dots \leq \lambda_{|V|} \leq 2$ of \mathcal{N} are all real, non-negative and contained in $[0, 2]$. The smallest eigenvalue is 0 and its multiplicity is the same as the number of connected components. Assuming that the graph is connected (i.e. one connected component), one can show the following relationship with the graph conductance:

Lemma 1. [Spectral bound [28]] *The temporal graph conductance of a weighted graph can be bounded as follows: $\phi(G^{[t,t']}) = \eta(t, t') \lambda_2 / 2 \leq \phi(G^{[t,t']})$.*

Note that the above bound is valid for arbitrary weighted graphs, although we explicitly state it in the context of aggregated graphs including the normalization based on $\eta(t, t')$. The conductance of any approximate solution $\bar{\phi}$ can serve as an upper bound to that of the lowest conductance in $G(V, E, W)$ and can be employed to prune irrelevant intervals:

Corollary 4. [Pruning] *If $\bar{\phi} \leq \eta(t, t') \lambda_2(\mathcal{N}^{t,t'}) / 2$, then $[t, t']$ does not contain the lowest conductance temporal community.*

The corollary follows directly from the spectral bound. An intuitive approach for pruning is to compute $\eta(t, t') \lambda_2(\mathcal{N}^{t,t'}) / 2$ for the aggregated graphs of all possible intervals in time, incurring a quadratic number of eigenvalue computations which will not scale to large graphs evolving over long periods of time. In what follows, we show that one can obtain a lower bound for λ_2 of an interval based on the eigenvalues in sub-intervals reducing the number of necessary eigenvalue computations in our pruning strategy.

Lemma 2. *Let A be a real positive semi-definite matrix of dimension n , and let d and ϵ be two real vectors of the same dimension s.t. $d_i \geq 0, \epsilon_i \geq 0 \forall i \leq n$. Then, $\min_{f \perp d + \epsilon} f^T A f \geq \min_{g \perp d} g^T A g$.*

Proof. Available in the Appendix. \square

Theorem 5. [Composite bound] *Let $[t, t']$ be partitioned in k consecutive non-overlapping subintervals $\{[t_1, t'_1], [t_2, t'_2] \dots [t_k, t'_k]\}$ such that $t_i = t'_{i-1} - 1, \forall i \in [1, k]$ with corresponding aggregated normalized graph Laplacians \mathcal{N}_i . Then,*

$$\lambda_2(\hat{\mathcal{N}}) \geq \sum_{i=1}^k \min_{u \in V} \frac{\text{vol}(u, t_i, t'_i)}{\text{vol}(u, t, t')} \lambda_2(\mathcal{N}_i),$$

where $\lambda_2(\mathcal{N}_i)$ is the second smallest eigenvalue of \mathcal{N}_i , and $\hat{\mathcal{N}}$ is the Laplacian of $G^{[t,t']}$.

Proof. Available in the Appendix. \square

The composite bound for $\lambda_2(\hat{\mathcal{N}})$ enables pruning intervals without explicitly computing their interval eigenvalues. Given any partitioning $\{[t_1, t'_1] \dots [t_k, t'_k]\}$ of $[t, t']$, we can prune using the composite bound $\phi_c(G^{[t,t']}) = \eta(t, t') \sum_{i=1}^k \min_{u \in V} \frac{\text{vol}(u, t_i, t'_i)}{\text{vol}(u, t, t')} \lambda_2(\mathcal{N}_i)$.

To enable scalable pruning, we can pre-compute λ_2 for a subset of intervals and attempt to prune all intervals using

ϕ_c instead of an exhaustive eigenvalue computation. There is a trade-off between how many eigenvalues to pre-compute and the pruning power of ϕ_c . If we only compute single-time snapshot intervals, we can obtain all composite bounds, however, they may not be very tight for longer intervals. If we pre-compute too many intervals, we will incur cost similar to the exhaustive all-eigenvalue computation.

We adopt a multi-scale scheme in which we pre-compute non-overlapping intervals of exponentially increasing lengths:

$$l^i, l \in \mathbb{N}_{\geq 2}, \forall i \in \mathbb{N}_{[0, \lceil \log(T) \rceil]}.$$

For example, if $l = 2$, we compute λ_2 for non-overlapping intervals of sizes powers of 2, i.e. $\{[0, 0], \dots [T, T], [0-1], [2-3], \dots [T-1, T], \dots\}$. The pre-computation requirement for our composite bound scheme is $O(T \log(T) B)$, where B is the time to compute λ_2 for a single aggregated graph using the Lanczos method. To compute ϕ_c for any interval we incur cost $O(|E| \log(T))$ as any interval can be composed by at most $O(\log(T))$ subintervals with known eigenvalues.

While our composite scheme requires sub-quadratic (in T) eigenvalue computations, we still need to compute ϕ_c for $O(T^2)$ intervals to prune them. To further speed up the process, we group intervals of significant overlap and attempt to prune using a group-level bound without composing individual intervals within the group. To this end, we define a *pruning group* $\tau = (t, t', t'')$ as a set of intervals with a common start t and ending at times t' to t'' , $t' < t''$. We ensure a significant overlap between all group members by enforcing that the common interval prefix exceeds a fixed fraction of the length of all group members: $\frac{t' - t}{t'' - t} \geq \beta$. Given a partitioning $\{[t_i, t'_i]\}$ of the group prefix $[t, t']$, we define the group lower bound as $\phi_c(G^\tau) = \eta(t, t'') \sum_{i=1}^k \min_{u \in V} \frac{\text{vol}(u, t_i, t'_i)}{\text{vol}(u, t, t'')} \lambda_2(\mathcal{N}_i)$. The differences from the composite bound of the prefix $\phi_c(G^{[t,t']})$ is in (i) the denominator of the fraction and (ii) the normalization $\eta(t, t'')$ on the RHS.

Theorem 6. [Group composite bound] *Let $\tau = (t, t', t'')$ be a group of shared-prefix intervals, then $\phi_c(G^\tau) \leq \phi_c(G^{[t,t'']})$, $\forall t^* \in [t', t'']$.*

Proof. Available in the Appendix. \square

D. PHASR: Prune, HASH and Refine

The steps of our overall method PHASR are detailed in Alg. 1. We first pre-compute the eigenvalues for a set Φ of $O(T)$ intervals as outlined in Sec. IV-C (Step 1) and find an estimate ϕ^* of the solution by probing a constant number of promising periods of small λ_2 in Φ (Step 2). We employ a light-weight version of hashing in those periods. Then we prune groups by composing their bounds $\phi_c(G^\tau)$ based on Φ (Step 3), and for unpruned groups we compute composite bounds of individual intervals and attempt to prune them (Step 4). We next hash neighborhoods N_u^t of nodes, targeting all possible scales s for collision that include unpruned intervals (Steps 5-11). To target a particular time scale s , we select the appropriate number of time pivots $k^*(s)$ according to Thm. 2 (Step 8). Next, we process collision buckets B containing sets of neighborhoods N_u^t ordered by a decreasing fill-factor, quantifying the consistency

Algorithm 1: PHASR

Require: $G(V, E, W)$, α , LSH rows r , bands b , pruning res. l
Ensure: A set of temporal communities $\mathcal{C} = \{(C_i, t_i, t'_i)\}$

- 1: Compute bounds Φ at scales $l^i, i = 0 \dots \lceil \log(T) \rceil$
- 2: Compute an estimate ϕ^* using Φ
- 3: Prune intervals $[t, t'] \in \tau$ based on $\phi_{\mathcal{C}}(G^\tau) \geq \phi^*$
- 4: Prune remaining intervals $[t, t']$ based on $\phi_{\mathcal{C}}(G^{[t, t']}) \geq \phi^*$
- 5: **for all** $(u, t) \in (V, [1 \dots T])$ **do**
- 6: **for all** scales $s \in 1 \dots T/2$ **do**
- 7: **if** \exists an unpruned $[l, r] \in [t - s, t + s]$, **then**
- 8: $Hash(N_u^t, k^*(s), r, b)$
- 9: **end if**
- 10: **end for**
- 11: **end for**
- 12: **for** \forall Buckets B sorted by *fill-factor* **do**
- 13: $[l, r] =$ interval of B
- 14: **if** $\phi_{\mathcal{C}}(G^{[l, r]}) < \phi^*$ **then**
- 15: $(C, t, t') = Refine(B)$
- 16: $\phi^* = \min(\phi^*, \phi(C, t, t'))$
- 17: Add (C, t, t') to \mathcal{C}
- 18: **end if**
- 19: **end for**
- 20: **RETURN** \mathcal{C}

of node sets in the timestamps with the bucket (Steps 12-19). If the interval spanned by the current bucket’s timestamps cannot be pruned, we form the aggregated graph $G^{[l, r]}$ and we compute the lowest temporal community C around the seed nodes in the bucket using the spectral sweep method by Andersen et al. [3] (Step 15); briefly, this operates by considering just the top-ranking node in the bucket, then the top two, then the top three, and so on. We maintain the best estimate in ϕ^* to enable more pruning of buckets to be processed (Step 16) and add C to the result set \mathcal{C} (Step 17). Finally, we report \mathcal{C} . Note that we can easily maintain and report multiple top communities in Steps 12-20.

Complexity analysis: Precomputing Φ requires $O(T)$ eigenvalue computations, since we consider non-overlapping intervals of exponentially increasing sizes. The group pruning requires $O(T \log^2 T |V|)$ time, since when ensuring overlap of at least $\beta < 1$ among interval group members in the grouping, we get $O(\log T)$ groups for every starting position and a total of $O(T \log T)$ groups. To compute the composite group bound we need at most $O(\log T)$ precomputed eigenvalue intervals and a scan over the node volumes, arriving at the final group pruning complexity. It is important to note that the eigenvalue computations, pruning and candidate verification can all be trivially parallelized on common MapReduce-like systems. The time spent in the remainder of the algorithm depends on the effectiveness of the pruning steps which, as we show in the evaluation, are able to filter most intervals given the existence of outstanding local temporal communities.

V. EXPERIMENTAL EVALUATION

We evaluate the quality and scalability of our approach in both synthetic and real-world networks. Of main interest are the running time savings due to the pruning enabled by our bounds and the quality of candidate communities produced by hashing. We conduct all experiments on a 3.6GHz Intel processor with 16GB of RAM. All algorithms are implemented

as single-thread Java programs. To compute eigenvalues, we employ the implementation of the Lanczos algorithm from the *Matrix Toolkit Java*¹

A. Datasets and competing techniques

Datasets: We use preferential attachment synthetic networks [5] of sizes between $1k$ and $15k$ nodes and average degree of 20. We replicate the unweighted network structure over $T = 1000$ timestamps and assign Poisson random weights with mean 5 on edges in time independently. We inject a strong temporal community (C, t, t') of length $t' - t = 10$ by increasing the average weight on the internal edges. The community *contrast* is defined as the ratio of the mean weight in (C, t, t') and the global average of 5; we used a contrast value of 8 in synthetic data unless otherwise specified.

We also use real-world datasets of various length, number of nodes and density listed in Tab. I. The *Road* traffic dataset is a subnetwork of the California highway system. Edge (road segments) are weighted based on the average speed at $5m$ intervals. In this dataset we aim to detect contiguous subnetworks of abnormal speeds over time. To detect high- and low-speed temporal subgraphs we assign weights as \mathcal{V}^2 or $(85 - \mathcal{V})^2$ respectively, where \mathcal{V} is the speed in mph at a given time. Execution times for both weighting schemes are similar. The *Internet* traffic data is a $2h$ trace of all p2p web traffic at the level of organizations (first three bytes of host IPs) from June 2013 on a backbone link in Japan, where weights are assigned as the number of packets between a pair of organizations at $1m$ resolution [10]. Our densest dataset is a *Call* graph among sectors of the city of Milan, Italy over $24h$ period [12]. Edge weights correspond to the number of calls between sectors within an hour.

A note on data sizes and parallelization opportunities: It is important to note that while the real-world networks we employ for experimentation are in the order of thousands of nodes, the search space over all possible intervals requires consideration of $O(T^2)$ differently weighted graphs of that size. For example, in our *Call* dataset all-interval graphs contain cumulatively *208 million edges* while in the largest Synthetic dataset the cumulative number edges exceeds *1.4 billion*. This large search space is the reason why exhaustive baselines do not scale to instances of such sizes. In addition, our technique can be easily parallelized employing a bulk synchronous processing system (BSP) such as Hadoop², since the underlying building blocks of eigenvalue computations, hashing and aggregation of colliding neighborhoods fit naturally the BSP programming paradigm. A parallel implementation and corresponding scalability experiments are beyond the scope of this work, but we plan to include them in an extended version of this work.

Baselines: We compare PHASR to three baselines: (1) exhaustive (*EXH*) temporal extension of the spectral sweep method by Andersen et al. [3]; (2) a temporal extension of the hashing community detection (*H+RW*) [23]; and (3) the

¹Matrix Toolkit Java from <https://github.com/fommil/matrix-toolkits-java>.

²Hadoop. <http://hadoop.apache.org/>

Dataset	V	E	T	PHASR		EXH [3]*		H+RW [23]*		L-metric [29]	
				Time	ϕ	Time	ϕ	Time	ϕ	Time	ϕ
Synth.	1k-15k	20k-300k	1k	76s	0.017	days	n/a	days	n/a	6486s	0.042
Road	100	128	1k	118s	0.039	days	n/a	> 24h	n/a	1s	0.099
Internet	2542	12699	120	2.3h	0.008	days	n/a	> 24h	n/a	> 6h	n/a
Call	1333	756k	24	28m	0.0032	days	n/a	> 24h	n/a	3.5h	0.215

TABLE I

DATA SETS USED FOR EXPERIMENTATION AND COMPARISON TO THE L-METRIC DYNAMIC COMMUNITY METHOD [29]. *COMPARISONS TO OTHER BASELINES WERE INFEASIBLE ON THE FULL DATASETS. SEE FIG. 4 FOR RESULTS ON SMALLER VERSIONS OF THESE DATASETS.

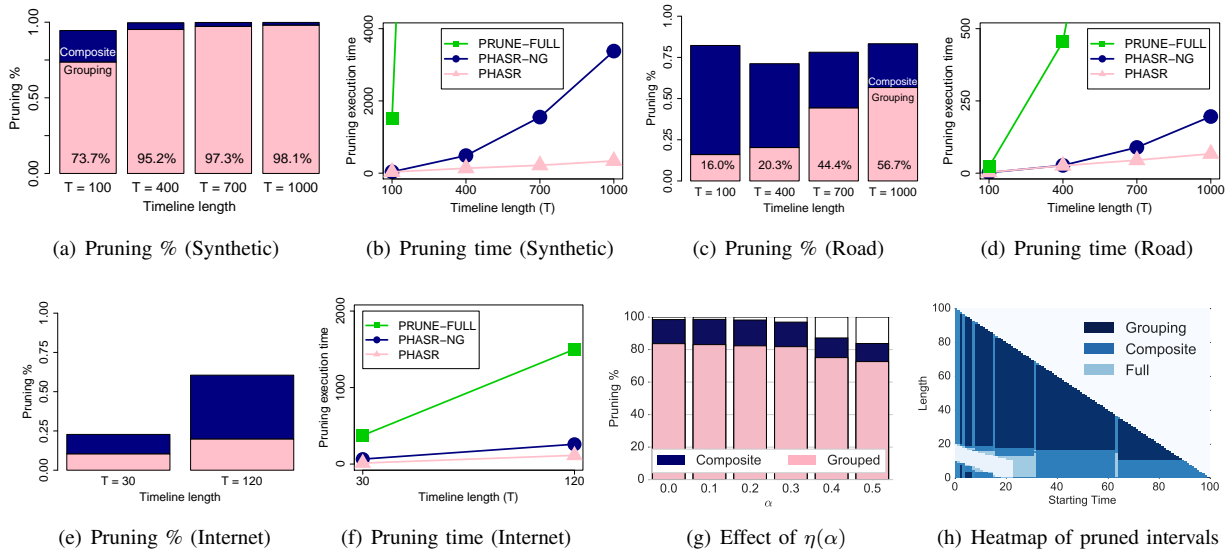


Fig. 3. (a) Percentage of all pruned intervals using the group and then composite pruning in PHASR. (b) Comparison of the time taken for pruning with (PHASR) and without grouping (PHASR-NG), and full calculation of all Cheeger bounds (PRUNE-FULL). Similar comparisons for the Internet (e),(f) and Road (c),(d) networks. (g) Pruning power in synthetic data for different levels of temporal normalization α . (h) Heatmap representation of the pruned intervals in one of the synthetic datasets for $T = 100$ where each pixel represents an interval with horizontal coordinate its starting point and vertical coordinate its duration. Colors encode the kind of bound that enabled pruning the specific interval.

incremental *L-Metric* for local communities in dynamic graphs by Takaffoli et al.[29]. *EXH* performs spectral sweeps in all possible intervals and starting from all nodes. *H+RW* hashes neighborhoods in the graphs of all possible time intervals (thus can be viewed as naive dynamic extension of [23]), and then performs a sweep from seeds identified by hashing. Hashing candidates do not form low-conductance communities on their own since [23] does not consider cuts. *L-Metric* [29] incrementally extends local communities in time, by using the connected components of communities from the previous time step as seeds. Since it allows communities to change over time, we implement a post-processing step in which we maintain the largest node intersections for all possible intervals of a contiguous community in order to obtain dynamic communities of fixed membership. We also consider two versions of PHASR: explicit computation of all interval bounds for pruning, termed (*PRUNE-FULL*) and our method using composite and group bounds PHASR. We evaluate the pruning power for different pruning strategies, the scalability of competing techniques and the effect of parameters for PHASR in what follows.

B. Pruning power

We evaluate the pruning power of our bounds in both synthetic and real world datasets Fig. 3(a) shows the average pruning percentage of all intervals in the synthetic data for increasing T and an injected community of a fixed length of 10. We prune more than 95% of the possible intervals across

all lengths. The fast grouping phase prunes the majority of the intervals ranging from 73% when the injected community is 1/10-th of the timeline to more than 98% of the intervals for $T = 1000$. The execution time is presented in Fig. 3(b). Since most intervals are pruned at the group stage, PHASR's pruning time grows almost linearly with T , while the time for pruning based on composite bounds without grouping (PHASR-NG) grows faster, resulting in more than an order of magnitude savings at $T = 1000$ due to grouping. Computing all-interval eigenvalues (Full) does not scale beyond 100 time steps, requiring two orders of magnitude more time than composite and group pruning due to the expensive eigenvalue computations.

We perform similar pruning evaluation for the Road Figs. 3(c), 3(d) and Internet Figs. 3(e),3(f) datasets. We prune more than 75% of the intervals in Road and 55% in Internet. Grouping is less effective here as there are multiple temporal intervals with good temporal conductance, though still providing increased effectiveness reflected in the widening gap between the running times of PHASR and PHASR-NG for Road Fig.3(d). The savings of grouping are smaller for Internet as there exist communities of hosts of low conductance persisting over most of the 2h span. Exhaustive computation of eigenvalues, in comparison is at least an order of magnitude slower for the longest timespans of both datasets. The pruning percentage in the Call dataset is more than 75%

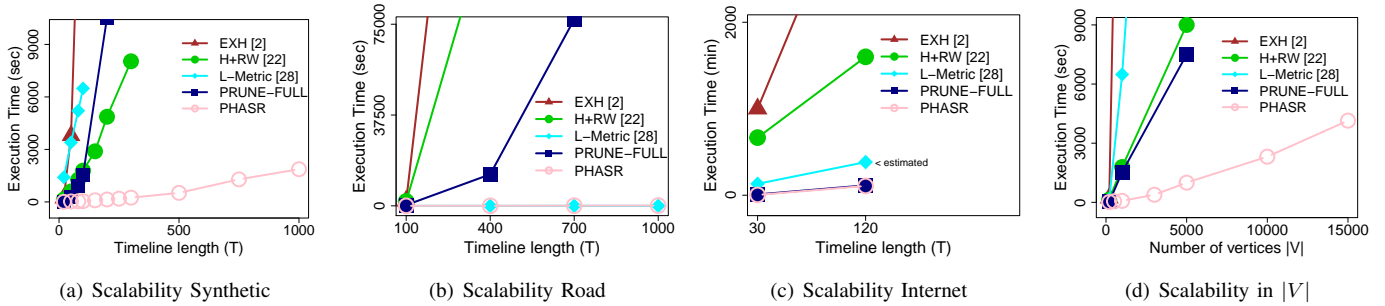


Fig. 4. Comparison of PHASR’s total running time with that of alternatives for increasing T in (a) Synthetic ($|V| = 1000$); (b) Road; and (c) the Internet datasets. (d): Scalability with the graph size ($T = 100$) on the Synthetic dataset.

when considering all 24 time intervals, allowing for fast overall time regardless of its density (no figure due to the short $T = 24$).

We also study the effect of temporal normalization on pruning effectiveness. The temporal normalization $\eta(t, t') = (t' - t)^{-\alpha}$ is controlled by the exponent parameter α , where higher values of α decrease the temporal conductance of longer intervals and thus make them more preferable. As we discussed earlier $\alpha = 0$ amounts to no normalization and in this case solutions tend to reside in a single timestamp. Alternatively, if α is very large solutions that span the whole timeline are preferred, though for such settings, simply aggregating the graph over all timestamps and employing static graph solutions will work better. We show the effect of targeting medium interval-length solutions (the most challenging setting) on the effectiveness of our pruning in Fig. 3(g). The total pruning decreases from 98% to 83% for values of α up to 0.5 and the fraction of both intervals pruned by the group bound (blue) and composite bound (red) reduce proportionally. Nevertheless, this level of pruning ensures significantly more efficient overall processing than employing hashing for all times and temporal scales in an exhaustive manner.

A detailed visualization of the pruning effectiveness in Synthetic is presented as a heatmap in Fig. 3(h). The space of all possible intervals is represented in a lower triangular matrix of pixels, where the pixel position encodes an interval start time (horizontal) and length (vertical axis). Grouping (darkest shade) prunes most of the long intervals and the majority of intervals of size less than 20 are pruned by the composite interval bound. Employing the Full eigenvalue computation may prune only a small percentage of additional intervals at the cost of lengthy eigenvalue computations (lightest shade). Intervals of significant overlap with the injected community (times 20-30) remain unpruned and considered for hashing.

C. Scalability.

PHASR scales well with increasing timeline length T . Particularly, its pruning phase is very efficient as evident in Figs. 3(b), 3(f), 3(d). The naïve approach of directly calculating all possible Cheeger bounds is quadratic and quickly becomes infeasible. The use of pruning groups here is key in Synthetic, Road and Call—while composite bounding without groups (PHASR-NG) is much better than the naïve approach, only

the full algorithm with pruning groups produces the near-linear scaling that is necessary for very long timelines.

The last 4 columns of Tab. 1 show the total running time and conductance of the best solutions for PHASR and L-Metric for the full datasets. In all cases except the Road dataset, PHASR completes much faster than the L-Metric: $10x$, $> 6x$ and $7x$ faster for Synthetic ($|V| = 1k$), Internet and Call respectively. L-Metric does not complete in more than 6 hours on Internet ($T = 120$). In all cases in which *L-Metric* completes, the discovered communities are of significantly worse conductance: 2.5 times worse in Synthetic and Road, and 67 times worse in Call. The reason for this lower quality is that L-Metric considers only adjacent time steps when trying to reconcile communities in time, while PHASR considers the full evolution of the graph at different scales.

Fig. 4 shows the complete running time of PHASR algorithm—pruning, hashing, and refinement via random walks—versus competing techniques over increasing T (Figs. 4(a), 4(b), 4(c)). Large numbers of vertices or time periods quickly render the exhaustive competitor methods infeasible, with runtimes of many hours or even days (estimated). The pruning and hashing segments of our approach scale well in both time and graph size; for large T , only about two percent of total execution time is spent on pruning, and the remainder on hashing and refinement. It is important to note that the running time of hashing and refinement can be reduced by considering smaller number of hash functions and bands at the expense of possibly worse-conductance results. A faster push-based local RW estimation, as described in [3], will enable scaling to larger network sizes as well. Our current implementation features only a naive full-network RWR, since scalable implementation of spectral sweeps is not the main focus of this work. L-Metric’s running time grows quickly with T and the graph size (Fig. 4(d)) and is dominated by our approach on all datasets, but the small Road dataset in which it completes faster, but discovers a worse community (Tab. 1). PHASR equipped with group pruning is the only alternative that scales with the graph size (Fig. 4(d)) and can be further improved by trivial parallel implementation as discussed earlier.

D. Case studies: Call and Internet

We obtained several low-conductance communities in the Milan telecom (*Call*) data; their locations are shown in Fig. 2. Multiple small communities formed at various times during the day in the circled region; their size suggest they may simply be coincidental. However, a larger, more coherent community was discovered in the area marked with a rectangle. Because it appears near the A50 highway during the early morning hours, we speculate it may be the result of calls from vehicles in commuter traffic.

The dominant community in the Internet traffic data covered much of the timeline we examined. The nodes involved all represented major telecom companies in a variety of countries: Japan, Saudi Arabia, Korea, Israel, and the United Kingdom; the low conductance here appears to be because of extremely high traffic between what we speculate are backbone Internet providers.

E. Effect of parameters

We also evaluate the effect of hashing parameters on the quality of obtained seeds. Our experiment demonstrate that higher number of hashing bands and neighborhood hashing functions increases the quality of obtained seeds; however, we observe diminishing returns past 7 bands in Synthetic. Details omitted due to space limitations.

VI. CONCLUSIONS

We proposed the problem of local temporal communities with the goal of detecting a subset of nodes and a time interval in which the nodes interact exclusively with each other. We generalized the measure of conductance to the temporal context and proposed a method PHASR for the minimum conductance temporal community. To scale the search in time we employed a novel spectral pruning approach that is sub-quadratic in the length of the total timeline. To scale the search in the graph space we proposed a time-and-graph locality sensitive family for neighborhoods of nodes which effectively spots cores of good communities in time.

We evaluated PHASR on both real and synthetic datasets and demonstrated that it scales better than alternatives to large instances, achieving two orders of magnitude running time reduction in Synthetic and 3 to 7 times reduction on big real instances compared to alternatives. PHASR also discovered communities of 2 to 67 times lower conductance than those obtained by a dynamic community baseline from the literature. This performance and accuracy is enabled by pruning as much as 95% of the possible time intervals in Synthetic and between 55% and 75% in real-world datasets; and due to our effective temporal hashing scheme for spotting good seeds in unpruned intervals.

REFERENCES

- [1] Rezwan Ahmed and George Karypis. Algorithms for mining the evolution of conserved relational states in dynamic networks. In *ICDM*, 2011.
- [2] Reid Andersen and Yuval Peres. Finding sparse cuts locally using evolving sets. In *Proceedings of the*

- Forty-first Annual ACM Symposium on Theory of Computing*, STOC '09, pages 235–244, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-506-2. doi: 10.1145/1536414.1536449. URL <http://doi.acm.org/10.1145/1536414.1536449>.
- [3] Reid Andersen, Fan Chung, and Kevin Lang. Local graph partitioning using pagerank vectors. In *Foundations of Computer Science, 2006. FOCS'06. 47th Annual IEEE Symposium on*, pages 475–486. IEEE, 2006.
- [4] Haim Avron and Lior Horesh. Community Detection Using Time-Dependent Personalized PageRank. In *ICML*, 2015.
- [5] Albert-László Barabási and Réka Albert. Emergence of scaling in random networks. *Science*, 1999.
- [6] Tanya Berger-Wolf, Chayant Tantipathananandh, and David Kempe. Dynamic community identification. In *Link Mining: Models, Algorithms, and Applications*, pages 307–336. Springer New York, 2010. doi: 10.1007/978-1-4419-6515-8_12.
- [7] Petko Bogdanov, Misael Mongiovi, and Ambuj K. Singh. Mining heavy subgraphs in time-evolving networks. In *ICDM*, 2011.
- [8] Ulrik Brandes, Daniel Delling, Marco Gaertler, Robert Görke, Martin Hoefer, Zoran Nikoloski, and Dorothea Wagner. Maximizing modularity is hard. *arXiv preprint physics/0608255*, 2006.
- [9] Andrei Z Broder, Moses Charikar, Alan M Frieze, and Michael Mitzenmacher. Min-wise independent permutations. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 327–336. ACM, 1998.
- [10] Kenjiro Cho, Koushirou Mitsuya, and Akira Kato. Traffic data repository at the wide project. In *Proceedings of the Annual Conference on USENIX Annual Technical Conference*, ATEC '00, Berkeley, CA, USA, 2000. USENIX Association.
- [11] Fan RK Chung. *Spectral graph theory*, volume 92. American Mathematical Soc., 1997.
- [12] Dandelion. Open big data. <https://dandelion.eu/datamine/open-big-data/>.
- [13] Santo Fortunato. Community detection in graphs. *Physics Reports*, 486(3-5):75 – 174, 2010. ISSN 0370-1573.
- [14] Noé Gaumont, Clémence Magnien, and Matthieu Latapy. Finding remarkably dense sequences of contacts in link streams. *Social Network Analysis and Mining*, 6(1):87, 2016.
- [15] S. Goel, A. Baykal, and D. Pon. Botnets: the anatomy of a case. *Journal of Information Systems Security*, 2006.
- [16] Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 604–613. ACM, 1998.
- [17] Sergey Ioffe. Improved consistent sampling, weighted minhash and l1 sketching. In *Data Mining (ICDM), 2010. Tenth IEEE International Conference on*, pages 246–255.

IEEE, 2010.

- [18] Min-Soo Kim and Jiawei Han. A particle-and-density based evolutionary clustering method for dynamic networks. *VLDB Endow.*, 2, 2009.
- [19] Jon Kleinberg. Bursty and hierarchical structure in streams. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '02, pages 91–101, New York, NY, USA, 2002. ACM. ISBN 1-58113-567-X. doi: 10.1145/775047.775061. URL <http://doi.acm.org/10.1145/775047.775061>.
- [20] Jure Leskovec, Kevin J Lang, and Michael Mahoney. Empirical comparison of algorithms for network community detection. In *Proceedings of the 19th international conference on World wide web*, pages 631–640. ACM, 2010.
- [21] Yu-Ru Lin, Yun Chi, Shenghuo Zhu, Hari Sundaram, and Belle L. Tseng. Facetnet: a framework for analyzing communities and their evolutions in dynamic networks. In *WWW*, 2008.
- [22] Siyuan Liu, Shuhui Wang, and Ramayya Krishnan. Persistent community detection in dynamic social networks. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 78–89. Springer, 2014.
- [23] Kathy Macropol, , and Ambuj Singh. Scalable discovery of best clusters on large graphs. In *Proceedings of the VLDB Endowment*, pages 693–702. VLDB, 2010.
- [24] Giulio Rossetti, Luca Pappalardo, Dino Pedreschi, and Fosca Giannotti. Tiles: an online algorithm for community discovery in dynamic social networks. *Machine Learning*, pages 1–29, 2016.
- [25] Polina Rozenshtein, Nikolaj Tatti, and Aristides Giannis. Discovering dynamic communities in interaction networks. In *Proceedings of ECML/PKDD*, 2014.
- [26] Umang Sharan and Jennifer Neville. Temporal-relational classifiers for prediction in evolving domains. In *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*, pages 540–549. IEEE, 2008.
- [27] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(8):888–905, 2000.
- [28] Daniel a Spielman. Algorithms, Graph Theory, and Linear Equations in Laplacian Matrices. *International Congress of Mathematicians*, 1(2):1–23, 2010. ISSN 0308-1087.
- [29] Mansoureh Takaffoli, Reihaneh Rabbany, and Osmar R Zaiane. Incremental local community identification in dynamic social networks. In *Proceedings of the 2013 IEEE/ACM international conference on advances in social networks analysis and mining*, pages 90–94. ACM, 2013.
- [30] Jiří Šíma and Satu Elisa Schaeffer. On the np-completeness of some graph cluster measures. In *Proceedings of the 32Nd Conference on Current Trends in Theory and Practice of Computer Science*, SOFSEM'06, pages 530–537, Berlin, Heidelberg, 2006. Springer-

Verlag. ISBN 3-540-31198-X, 978-3-540-31198-0. doi: 10.1007/11611257_51. URL http://dx.doi.org/10.1007/11611257_51.

- [31] Yen-Chuen Wei and Chung-Kuan Cheng. Towards efficient hierarchical designs by ratio cut partitioning. In *Computer-Aided Design, 1989. ICCAD-89. Digest of Technical Papers., 1989 IEEE International Conference on*, pages 298–301. IEEE, 1989.
- [32] Jierui Xie, Stephen Kelley, and Boleslaw K Szymanski. Overlapping community detection in networks: The state-of-the-art and comparative study. *ACM Computing Surveys (csur)*, 45(4):43, 2013.
- [33] Weiren Yu, Charu C Aggarwal, Shuai Ma, and Haixun Wang. On anomalous hotspot discovery in graph streams. In *Data Mining (ICDM), 2013 IEEE 13th International Conference on*, pages 1271–1276. IEEE, 2013.

APPENDIX: PROOFS OF THEORETICAL RESULTS.

Proof of Theorem 1: Temporal locality.

Proof. Consider two timestamps, $t \leq t'$. Their hash values with respect to a k -partitioning $\tau^k(t)$ and $\tau^k(t')$ will differ only when $\exists p_i | t < p_i \leq t'$. Since every pivot p_i was chosen uniformly at random from the full timeline, the probability that it falls between t and t' is proportional to the duration between the timepoints divided by the timeline length: $\frac{t'-t}{T}$. The probability that a specific pivot is not selected between the two timepoints is $1 - \frac{t'-t}{T}$. The two hash values match if none of the k pivots is selected between them, and the probability of this event is $(1 - \frac{t'-t}{T})^k$ since all p_i are chosen independently. When $t' - t \leq \Delta_1$ this probability is greater than $(1 - \frac{\Delta_1}{T})^k$, and the analogous relationship holds for $t' - t \geq \Delta_2$. \square

Proof of Theorem 2: Optimal Number of pivots.

Proof. Observe that the probability of a pivot landing in a perfect bracketing position (on the left or the right) is $\frac{1}{T}$. The probability of landing anywhere outside our target community is $1 - \frac{\Delta^*}{T}$. Any of the $\binom{k}{2}$ pairs of pivots could be the "bookends", and these could occur in either order (left-right or right-left). Thus, the probability of a perfect partition is:

$$2 \binom{k}{2} \left(\frac{1}{T}\right)^2 \left(1 - \frac{\Delta^*}{T}\right)^{k-2} = \left(\frac{1}{T}\right)^2 \left(1 - \frac{\Delta^*}{T}\right)^{k-2} (k^2 - k).$$

Since we want to find the k that maximizes the probability of perfect partition, we find $\frac{\partial p}{\partial k}$:

$$\left(1 - \frac{\Delta^*}{T}\right)^{k-1} \left[(2k-1) \left(1 - \frac{\Delta^*}{T}\right)^{k-1} + (k^2 - k) \log\left(1 - \frac{\Delta^*}{T}\right) \right]$$

Since $0 < \frac{\Delta^*}{T} < 1$, one of the roots of $\frac{\partial p}{\partial k} = 0$ is strictly smaller than 1, and the only feasible solution is:

$$k^* = \frac{\log\left(1 - \frac{\Delta^*}{T}\right) - 2 - \sqrt{\log^2\left(1 - \frac{\Delta^*}{T}\right) + 4}}{2 \log\left(1 - \frac{\Delta^*}{T}\right)}.$$

This is well approximated by $\frac{2T}{\Delta^*}$ for $\frac{2T}{\Delta^*} \in [0, 1]$. \square

Proof of Lemma 2.

Proof. We have a convex minimization function over a convex set: $\min f^T A f$, such that $f \perp (d + \epsilon)$. The Lagrange form is: $L(f, \lambda) = f^T A f + \lambda f^T (d + \epsilon)$. Setting the gradient w.r.t.

f to zero gives us: $\frac{\partial L}{\partial f} = 2Af + \lambda(d + \epsilon) = 0 \Rightarrow f^* = -\frac{\lambda}{2}A^{-1}(d + \epsilon)$. Note that A^{-1} exists since A is positive semi-definite. By an analogous development, $g^* = -\frac{\lambda}{2}A^{-1}d$. Next we substitute f^* into the objective to obtain the inequality of interest:

$$\begin{aligned} \min_{f \perp d + \epsilon} f^T A f &= f^{*T} A f^* \\ &= \frac{\lambda^2}{4} (A^{-1}(d + \epsilon))^T A (A^{-1}(d + \epsilon)) \\ &= \frac{\lambda^2}{4} (d + \epsilon)^T A^{-1} (d + \epsilon) \\ &= \frac{\lambda^2}{4} (d^T A^{-1} d + \epsilon^T A^{-1} d + d^T A^{-1} \epsilon + \epsilon^T A^{-1} \epsilon) \\ &\geq \frac{\lambda^2}{4} d^T A^{-1} d = g^{*T} A g^* = \min_{g \perp d} g^T A g \end{aligned}$$

□

Proof of Theorem 5: Composite bound.

Proof. Let D_i denote the degree matrix of $G^{[t_i, t'_i]}$ and \hat{D} the degree matrix of $G^{[t, t']}$. According to the *Min-max* theorem the second eigenvalue of \hat{N} can be characterized as the minimum of the Rayleigh quotient in the subspace orthogonal to the first eigenvector $\hat{d}^{1/2}$ [28]: $\lambda_2(\hat{N}) = \min_{g \perp \hat{d}^{1/2}} \frac{g^T \hat{N} g}{g^T g} = \min_{f \perp \hat{d}} \frac{f^T \hat{\mathcal{L}} f}{f^T \hat{D} f}$, where \hat{d} is a vector of the node volumes in $[t, t']$ and f and g are column vectors. The second equality is obtained by variable change $g = \hat{D}^{1/2} f$ and has the form of a generalized eigenvalue problem. We can then show:

$$\lambda_2(\hat{N}) = \min_{f \perp \hat{d}} \frac{f^T \hat{\mathcal{L}} f}{f^T \hat{D} f} = \min_{f \perp \hat{d}} \sum_{i=1}^k \frac{f^T \mathcal{L}_i f}{f^T \hat{D} f} \quad (1)$$

$$= \min_{f \perp \hat{d}} \sum_{i=1}^k \frac{f^T D_i f}{f^T \hat{D} f} f^T \mathcal{N}_i f \quad (2)$$

$$\geq \min_{f \perp \hat{d}} \sum_{i=1}^k \frac{f^T D_i f}{f^T \hat{D} f} \lambda_2(\mathcal{N}_i) \quad (3)$$

$$= \min_{g \perp \hat{d}^{1/2}} \sum_{i=1}^k \frac{g^T (\hat{D}^{-1/2} D_i \hat{D}^{-1/2}) g}{g^T g} \lambda_2(\mathcal{N}_i) \quad (4)$$

$$= \min_{g \perp \hat{d}^{1/2}} \sum_{i=1}^k \frac{\sum_{u \in V} g_u^2 \frac{vol(u, t_i, t'_i)}{vol(u, t, t')}}{\|g\|^2} \lambda_2(\mathcal{N}_i) \quad (5)$$

$$\geq \sum_{i=1}^k \min_{u \in V} \frac{vol(u, t_i, t'_i)}{vol(u, t, t')} \lambda_2(\mathcal{N}_i) \quad (6)$$

For (1) we use the fact that $\hat{\mathcal{L}} = \sum_{i=1}^k \mathcal{L}_i$. In (2) we multiply all summands by $f^T D_i f / f^T D_i f$. (3) follows from Lem. 2 and in (4) we have substituted back the vector variables $g = \hat{D}^{1/2} f$. In (5) we have applied the definitions of the sub-interval volume for nodes and have unfolded the quadratic form for the diagonal matrix $g^T (\hat{D}^{-1/2} D_i \hat{D}^{-1/2}) g$. Finally, since:

$$\sum_{u \in V} g_u^2 \frac{vol(u, t_i, t'_i)}{vol(u, t, t')} \geq \|g\|^2 \min_{u \in V} \frac{vol(u, t_i, t'_i)}{vol(u, t, t')},$$

we obtain the inequality in (6). □

Proof of Theorem 6: Group composite bound.

Proof. Let t^* be the end-point of one of the subintervals in the group, i.e. $t \leq t' \leq t^* \leq t''$. Then, since $\eta()$ is monotonically decreasing as long as $\alpha \geq 0$, we have $\eta(t, t^*) \geq \eta(t, t'')$. In addition, since the aggregated weights in super-intervals dominate those in sub-intervals we have that: $\min_{u \in V} \frac{vol(u, t_i, t_i^*)}{vol(u, t, t^*)} \geq \min_{u \in V} \frac{vol(u, t_i, t_i^*)}{vol(u, t, t'')}$. Using the above we have:

$$\begin{aligned} \underline{\phi}_c(G^{[t, t^*]}) &= \eta(t, t^*) \sum_{i=1}^k \min_{u \in V} \frac{vol(u, t_i, t_i^*)}{vol(u, t, t^*)} \lambda_2(\mathcal{N}_i) \\ &\geq \eta(t, t'') \sum_{i=1}^k \min_{u \in V} \frac{vol(u, t_i, t_i^*)}{vol(u, t, t'')} \lambda_2(\mathcal{N}_i) \\ &= \underline{\phi}_c(G^T) \end{aligned}$$

□