Local Community Detection in Dynamic Networks

Daniel J. DiTursi∗†
dditursi@siena.edu

Gaurav Ghosh∗
gghosh@albany.edu

Petko Bogdanov∗
pbogdanov@albany.edu

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

†Department of Computer Science
Siena College
Loudonville, NY 12211

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? 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 [1]. Similarly, prior to a major project deadline, members of a team may communicate more and exclusively among each other compared to other times [2]. 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 internal and weaker external interactions. 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][4], where locality is enforced in both (i) the graph domain: local as opposed to complete partitioning; and (ii) the time domain: bursty internal interactions in a contiguous interval.

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. 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][5] may occlude dynamic communities due to mixing interactions from different periods, while examination of single 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. We spot candidate community nodes in unpruned time intervals using a time-and-graph-aware locality sensitive hashing scheme to generate likely community seeds. In the refinement step, we expand the seeds to communities in time. Our contributions in this work are as follows:

- We propose novel spectral bounds for temporal 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 [5][6][7][8] in that we consider a dy-
Dense subgraph detection has been recently considered for unweighted interactions in time [9][10][11]. These methods require the desired community length as a parameter and do not consider the isolation of the target community (the cut).

Persistent subgraphs in time have also been examined [12][13]. While they have a temporal component, the objective in these works is somewhat orthogonal to our own, as they seek stable activity levels and do not address the cut.

High-weight temporal subgraph detection is another relevant setting in which the goal is detecting connected subgraphs which optimize a function of their node or edge weights in time [14][15][16]. Once again, however, there is no consideration of the cut.

Evolutionary clustering for dynamic networks is a very active area of research [17][18][19]. The problem setting differs, though, in that the goal is to partition all node/time pairs into clusters instead of simply returning the best communities.

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 [20][3]. Our problem can then be cast as detecting the subgraph and interval of smallest temporal conductance.

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 $T = \{0,1,\ldots,[T]-1\}$ 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 $vol(u,t,t') = \sum_{v \in E} w(u,v,t,t')$. 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 [20], is defined as:

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

where $C = V \setminus C$; $cut(C,t,t') = \sum_{u \in C \cap \bar{C}} w(u,v,t,t')$ 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 $\alpha$ 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 [21], [22].

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: $$(C_o,t_o,t'_o) = \arg \min_{C \subseteq V, 0 \leq t \leq T} \phi(C,t,t').$$

The lowest conductance problem in a static graph is known to be NP-hard [20] 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 detection of community seeds in the temporal graph which can be used as input for approximate approaches in the static case [3], [4].

IV. METHODS

Our 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-A); and (ii) with the length of the timeline by pruning infeasible periods in time with guarantees (Pruning Sec. IV-B). Our overall approach is presented in Sec. IV-C.

A. Temporal Neighborhood LSH to spot seeds

A first step in our approach is to find seeds: node/time pairs that can then be used to expand to strong communities. We observe that, within good communities, weighted node neighborhoods tend to be similar. We exploit this observation by applying locality-sensitive hashing (LSH)[23] to node neighborhoods in time, adding a temporal component to the weighted minhash procedure developed by Ioffe et al.[24]

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

$$P[\psi(N^t_u) = \psi(N^t_v)] = JW(N^t_u, N^t_v).$$

We use $\psi$ independent minhash functions to create a signature $S^t_G(N^t_u)$ for a given neighborhood $N^t_u$.

Merely comparing weighted neighborhoods is insufficient—we need to associate highly similar neighborhoods that are also close in time. We create a temporal LSH function using this intuition: Close time instants are likely to belong to the same interval if the timeline is partitioned into random segments.

Let $p^k = \{p_1 < p_2 \cdots < p_k\}$ 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 $t$: $\tau^k(t) = \{\min(i)|p_i \geq t, p_i \in p^k\}$. 848
Theorem 1. [Temporal locality] The family of temporal hash functions $\phi^k$ is a $(\Delta_1, \Delta_2, (1 - \frac{2^k}{2^{k+1}})^k, (1 - \frac{2^k}{2^{k+1}})^k)$-sensitive family for the distance in time $\Delta$ defined as the delay between two timepoints. (Proof available in extended version [25].)

To detect similar neighborhoods in time we combine an $r$-sized graph signature $S^e_r(N'_u)$ with a temporal signature $S^t_r(N'_u) = \phi^t(t)$ using an AND predicate to obtain a unified temporal neighborhood signature $S^{r,k}(N'_u)$ that is a locality sensitive family in both the time and graph domains as a direct consequence of the locality $S^e_r(N'_u)$ and $S^t_r(N'_u)$.

Corollary 2. Let $S^{r,k}$ be a temporal neighborhood hash function with $r$ minhashes and $k$ partitions. Then: $P[S^{r,k}(N'_u) = S^{r,k}(N'_u)] = J_W(N'_u, N'_u)^r(1 - \frac{2^k}{2^{k+1}})^k$.

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 $1 - (1 - [1 - pr,k]^l)$, where $pr,k = P[S^{r,k}(N'_u) = S^{r,k}(N'_u)]$. Similarly, an AND predicate composition will result in $pr,k$ probability of collision. Using such compositions we can control the sensitivity and specificity of our LSH scheme.

B. Spectral bounds for pruning time intervals

In real-world graphs many time intervals contain no promising communities—i.e. no project deadline or spike in network traffic. Such periods of low community activity cannot coincide with the best temporal community. We develop lower bounds on the temporal conductance of any subgraph in a given time window. We then employ our bounds in combination with a solution estimate to deterministically prune irrelevant intervals in time, significantly improving the running time of our LSH-based approach.

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 $C \in V$: $\phi(G^{(t,t')}) = \min_{C \subseteq V} \phi(S(t,t'))$. 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 $D_u = \sum_v w(u,v,t,t')$ and 0 in all off-diagonal elements. The matrix $L = D - A$ is the unnormalized graph Laplacian, while the matrix $\mathcal{N} = D^{-1/2}LD^{-1/2}$ is the symmetric normalized graph Laplacian [26]. The Laplacian matrices have many advantageous properties and have been employed in spectral graph partitioning [26], [27]. The eigenvalues of $\mathcal{N}$ are all real, non-negative and contained in $[0,2]$. The smallest eigenvalue is $\lambda_1 = 0$ 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 dependence on the second-smallest eigenvalue, $\lambda_2$:

Lemma 1. [Spectral bound] 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 $\phi$ 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 3. [Pruning] If $\phi \leq \eta(t,t')\lambda_2(N^{t,t'}/2$, then $[t,t']$ does not contain the lowest conductance temporal community.

An intuitive approach for pruning is to compute $\eta(t,t')\lambda_2(N^{t,t'}/2$ for the aggregated graphs of all possible intervals in time, but this incurs 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 $\lambda_2$ of an interval based on the eigenvalues in sub-intervals, reducing the number of necessary eigenvalue computations in our pruning strategy.

Theorem 4. [Composite bound] Partition $[t,t']$ into consecutive non-overlapping subintervals $\{[t_1,t_1'],[t_2,t_2'],...,[t_k,t_k']\}$ such that $t_i = t_{i+1} - 1, \forall i \in [1,k]$ with corresponding aggregated normalized Laplacians $\mathcal{N}_i$. Then:

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

where $\lambda_2(\mathcal{N}_i)$ is the second smallest eigenvalue of $\mathcal{N}_i$, and $\lambda_2$ is the Laplacian of $G^{(t,t')}$. (Proof in [25].)

The composite bound for $\lambda_2$ enables pruning intervals without explicitly computing their interval eigenvalues. Given any partitioning $\{[t_1,t_1'],[t_2,t_2'],...,[t_k,t_k']\}$ of $[t,t']$, we can prune using the composite bound $\phi_\lambda(G^{(t,t')}) = \eta(t,t') \sum_{i=1}^k \min_{u \in V} \frac{\text{vol}(u,t_i,t_i')}{\text{vol}(u,t,t')} \lambda_2(\mathcal{N}_i)$.

To enable scalable pruning, we can pre-compute $\lambda_2$ for a subset of intervals and attempt to prune all intervals using $\phi_\lambda$ instead of an exhaustive eigenvalue computation. There is a trade-off between how many eigenvalues to pre-compute and the pruning power of $\phi_\lambda$. 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:

$$l_i, l \in \mathbb{N}_{\geq 2}, \forall i \in \mathbb{N}_{\lfloor \log(T/2) \rfloor}.$$
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: \( \frac{t’ - t}{t’} \geq \beta \). Given a partitioning \( \{[t_i, t_i']\} \) of the group prefix \([t, t']\), we define the group lower bound as \( \phi_c(G^t_{[t,t']}) = \eta(t, t’’) \sum_{k=1}^{m} \frac{1}{\max_{i \in V} \text{vol}(l, r, b) \lambda_2(N_i)} \). The differences from the composite bound of the prefix \( \phi_c(G^t_{[t,t']}) \) is in (i) the denominator of the fraction and (ii) the normalization \( \eta(t, t’’\). On the RHS.

**Theorem 5. [Group composite bound] Let \( \tau = (t, t’, t’’) \) be a group of shared-prefix intervals, then \( \phi_c(G^t_{[t,t']}) \leq \phi_c(G^t_{[t,t’’]), \forall \phi^* \in [t’, t’’}\). (Proof in [25].)

**C. 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 \( \Phi \) of \( O(T) \) intervals as outlined in Sec. IV-B (Step 1) and find an estimate \( \phi^* \) of the solution by probing a constant number of promising periods of small \( \lambda_2 \) in \( \Phi \) (Step 2). We employ a lightweight versioning of those hashtags. Then we prune groups by composing their bounds \( \phi_c(G^t) \) based on \( \Phi \) (Step 3), and for unpruned groups we compute composite bounds of individual intervals and attempt to prune them (Step 4). We next hash neighborhoods \( N^v_\Phi \) of nodes (Steps 5-11). Next, we process collision buckets \( B \) containing sets of neighborhoods \( N^v_\Phi \) ordered by a decreasing fill-factor, quantifying the consistency 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 \( G^t_{[t,t’]} \) and we compute the lowest temporal community \( C \) around the seed nodes in the bucket using [3] (Step 15). We maintain the best estimate in \( \phi^* \) to enable more pruning of buckets to be processed (Step 16) and add \( C \) to the result set \( C \) (Step 17). Finally, we report \( C \). Note that we can easily maintain and report multiple top communities in Steps 12-20.

**Algorithm 1: PHASR**

```
Require: \( G(V,E,W) \), \( \alpha \), LSH rows \( r \), bands \( b \), pruning res. \( l \)
Ensure: A set of temporal communities \( C = \{C(t_i, t_i')\} \)
1: Compute bounds \( \Phi \) at scales \( t_i, i = 0 \ldots \lfloor \log(T) \rfloor \)
2: Compute an estimate \( \phi^* \) using \( \Phi \)
3: Prune intervals \( [t_i, t_i'] \in \tau \) based on \( \phi_c(G^t_{[t_i, t_i']}) \geq \phi^* \)
4: Prune remaining intervals \( [t_i, t_i'] \) on the bucket based on \( \phi_c(G^t_{[t_i, t_i']}) \geq \phi^* \)
5: for all \((u, t) \in (V, \{1 \ldots T\}) do\)
6: for all scales \( s \in 1 \ldots T/2 \) do\)
7: if \( \exists \) an unpruned \([u, t] \in [2 - s, t + s] \), then
8: \( \text{Hash}(N^v_\Phi, k^s(s, r, b)) \)
9: end if
10: end for
11: end for
12: for \( \forall \) Buckets \( B \) sorted by fill-factor do
13: \([l, r] = \text{interval of } B \)
14: if \( \phi_c(G^t_{[l,r']}) \leq \phi^* \) then
15: \( (C, t, t') = \text{ReRun}(B) \)
16: \( \phi^* = \min(\phi^*, \phi(C, t, t')) \)
17: \( \text{Add} \{C, t, t'\} \to C \)
18: end if
19: end for
20: RETURN \( C \)
```

The prunning runtime is \( O(T \log^2 T |V|) \). The time spent in the remainder of the algorithm depends on the effectiveness of pruning; empirically, we are able to filter out most intervals.

**V. EXPERIMENTAL EVALUATION**

**A. Datasets and comparing techniques**

**Datasets:** We use preferential attachment [28] synthetic networks 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 5min 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 \( \sqrt{2} \) or \( (85 - \sqrt{2})^2 \) respectively, where \( \sqrt{V} \) 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 [29]. Our densest dataset is a Call graph among sectors of the city of Milan, Italy over 24h period [30]. Edge weights correspond to the number of calls between sectors within an hour.

**Baselines:** We compare PHASR to three baselines: (1) exhaustive (EXH) temporal extension of the spectral sweep method in [3]; (2) a temporal extension of hashing community detection (H+RW) [7]; and (3) the incremental L-Metric for local communities in dynamic graphs by Takaffoli et al.[19].

**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 [7]), and then performs a sweep from seeds identified by hashing. Hashing candidates do not form low-conductance communities on their own since [7] does not consider cuts. L-Metric [19] 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. In synthetic, we prune more than 95% of the possible intervals across all timeline lengths. The fast grouping phase prunes the majority of the intervals ranging from 73% when the injected community is 1/10 of the timeline to more than 98% of the intervals for 1/100. The execution time is presented in Fig. 2(a). 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 Fig. 2(b) and Internet Fig. 2(c) 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.2(b). 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% when considering all 24 time intervals, allowing for fast overall time regardless of its density.

A detailed visualization of the pruning effectiveness in Synthetic is presented as a heatmap in Fig. 2(d). 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, while the majority of intervals of size less than 20 are pruned by the composite bound. 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. We further analyze the pruning power and the effect of the normalization constant \( \alpha \) in the extended version of this paper[25].

C. Scalability.

PHASR scales well with increasing timeline length \( T \). Particularly, its pruning phase is very efficient as evident in Figs. 2(a),2(b),2(c). The naïve approach of directly calculating all possible Cheeger bounds is quadratic and quickly becomes infeasible. The use of pruning groups here is key—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. 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. 3 shows the complete running time of the PHASR algorithm—pruning, hashing, and refinement via random walks—versus competing techniques over increasing \( T \) (Figs. 3(a),3(b),3(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.
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. 3(d)) and can be further improved with a parallel implementation.

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, achieving two orders of magnitude running time reduction in synthetic and 3 to 7 times reduction on large real instances. 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 (a) pruning as many as 95% of the possible time intervals in synthetic and between 55% and 75% in real-world datasets and (b) effective temporal hashing for spotting good seeds in unpruned intervals.

REFERENCES

[1] S. Goel, A. Baykal, and D. Pon, “Botnets: the anatomy of a case,” Journal of Information Systems Security, 2006.
[2] J. Kleinberg, “Bursty and hierarchical structure in streams,” in KDD, 2002. [Online]. Available: http://doi.acm.org/10.1145/775047.775061
[3] R. Andersen, F. Chung, and K. Lang, “Local graph partitioning using pagerank vectors,” in FOCs, 2006.
[4] R. Andersen and Y. Peres, “Finding sparse cuts locally using evolving sets,” in STOC, 2009. [Online]. Available: http://doi.acm.org/10.1145/1536414.1536449
[5] S. Fortunato, “Community detection in graphs,” Physics Reports, vol. 486, no. 3-5, pp. 75 – 174, 2010.
[6] J. Xie, S. Kelley, and B. K. Szymanski, “Overlapping community detection in networks: The state-of-the-art and comparative study,” ACM Computing Surveys (csur), vol. 45, no. 4, p. 43, 2013.
[7] K. Macropol and A. Singh, “Scalable discovery of best clusters on large graphs,” in VLDB, 2016.
[8] J. Leskovec, K. J. Lang, and M. Mahoney, “Empirical comparison of algorithms for network community detection,” in WWW, 2010.
[9] P. Rozenshtein, N. Tatti, and A. Gionis, “Discovering dynamic communities in interaction networks,” in Proceedings of ECML/PKDD, 2014.
[10] N. Gaumont, C. Magnien, and M. Latapy, “Finding remarkably dense sequences of contacts in link streams,” Social Network Analysis and Mining, vol. 6, no. 1, p. 87, 2016.
[11] G. Rossetti, L. Pappalardo, D. Pedreschi, and F. Giannotti, “Tiles: an online algorithm for community discovery in dynamic social networks,” Machine Learning, pp. 1–29, 2016.
[12] S. Liu, S. Wang, and R. Krishnan, “Persistent community detection in dynamic social networks,” in PAKDD, 2014.
[13] R. Ahmed and G. Karypis, “Algorithms for mining the evolution of conserved relational states in dynamic networks,” in ICDM, 2011.
[14] P. Bogdanov, M. Mongiovi, and A. K. Singh, “Mining heavy subgraphs in time-evolving networks,” in ICDM, 2011.
[15] M. Mongiovi, P. Bogdanov, and A. K. Singh, “Mining evolving network processes,” in ICDM, 2013.
[16] M. Mongiovi, P. Bogdanov, R. Ranca, A. K. Singh, E. E. Papalexakis, and C. Faloutsos, “Netset: Spotting significant anomalous regions on dynamic networks,” in SDM, 2013.
[17] Y.-R. Lin, Y. Chi, S. Zhu, H. Sundaram, and B. L. Tseng, “Facnet: a framework for analyzing communities and their evolutions in dynamic networks,” in WWW, 2008.
[18] M.-S. Kim and J. Han, “A particle-and-density based evolutionary clustering method for dynamic networks,” VLDB Endow., vol. 2, 2009.
[19] M. Takaffoli, R. Rabban, and O. R. Zaiane, “Incremental local community identification in dynamic social networks,” in ASONAM, 2013.
[20] J. Šíma and S. E. Schaeffer, “On the np-completeness of some graph cluster measures,” in SOFSEM, 2006. [Online]. Available: http://dx.doi.org/10.1007/11611257_51
[21] U. Sharar and J. Neville, “Temporal-relational classifiers for prediction in evolving domains,” in ICDM, 2008.
[22] W. Yu, C. C. Aggarwal, S. Ma, and H. Wang, “On anomalous hotspot discovery in graph streams,” in ICDM, 2013.
[23] P. Indyk and R. Motwani, “Approximate nearest neighbors: Towards removing the curse of dimensionality,” in ICML, 2008.
[24] S. Ioffe, “Improved consistent sampling, weighted minhash and 1 sketching,” in ICDM, 2010.
[25] D. J. DiTursi, G. Ghosh, and P. Bogdanov, “Local community detection in dynamic networks,” arXiv preprint, 2017. [Online]. Available: https://arxiv.org/abs/1709.04033
[26] D. a. Spielman, “Algorithms, Graph Theory, and Linear Equations in Laplacian Matrices,” International Congress of Mathematicians, vol. 1, no. 2, pp. 1–23, 2010.
[27] F. R. Chung, Spectral graph theory. Am. Math. Soc., 1997, vol. 92.
[28] A.-L. Barabási and R. Albert, “Emergence of scaling in random networks,” Science, 1999.
[29] K. Cho, K. Mitsuya, and A. Kato, “Traffic data repository at the wide project,” in ATES, 2009.
[30] Dandelion, “Open big data. https://dandelion.eu/data/mining_open-big-data/.”