Efficient Algorithms for the Periodic Subgraphs Mining Problem

Alberto Apostolico\textsuperscript{a,b}, Péter L. Erdős\textsuperscript{c}, Ervin Győri\textsuperscript{c}, Zsuzsanna Lipták\textsuperscript{d}, Cinzia Pizzi\textsuperscript{b}

\textsuperscript{a}College of Computing, Georgia Institute of Technology, Atlanta, USA
\textsuperscript{b}Dipartimento di Ingegneria dell’Informazione, University of Padova, Italy
\textsuperscript{c}Alfréd Rényi Institute of Mathematics, Budapest, Hungary
\textsuperscript{d}AG Genominformatik, Technische Fakultät, Bielefeld University, Germany

Abstract

Given a sequence $G = \langle G_0, ..., G_{T-1} \rangle$ of simple graphs over uniquely labeled vertices from a set $V$, the periodic subgraph mining problem consists of discovering maximal subgraphs that recur at regular intervals in $G$. For a periodic subgraph to be maximal, it is intended here that it cannot be enriched by adding edges nor can its temporal span be expanded without losing support. We give faster algorithms than previously available for this problem. In particular, we show an optimal solution based on an implicit description of the output subgraphs that takes time $O(|V| + |\tilde{E}| \times T^2 / \sigma)$—where $|\tilde{E}|$ is the average number of edges over the entire sequence $G$—to publish all maximal periodic subgraphs that meet or exceed a minimum occurrence threshold $\sigma$.

Keywords: Discovery algorithms, sequences of networks, social networks, periodicity, interactions

1. Preliminaries

In recent years, graph theoretical methods have found increasing use in the social sciences, where individuals or entities interacting pairwise in a dynamic social network may be represented by the vertices of a graph, and
an interaction between any pair of individuals is represented by an edge. The evolution of such a network is then modeled as a sequence of graphs, in which edges over a set of uniquely labeled vertices evolve while a series of observations is made at regular intervals. In this context, it has become of interest to detect the emergence of various regularities, among which the periodically recurrent subgraphs that form the subject of the present paper. Apart from obvious contexts such as web communities, problems of this nature have emerged, e.g. in the study of mating patterns within communities of wild animals [6], in inferring predictable behaviors from sensor networks [3], etc.

Formally, by a dynamic network we understand a sequence $\mathcal{G} = \langle G_0, \ldots, G_{T-1} \rangle$ of simple graphs $G_t = (V_t, E_t)$, where each $V_t$ is a subset of the same, uniquely labeled vertex set $V$. The “dynamics” of the network is captured by the ever-changing edge sets $E_t$ over a series of time-steps $T = \{1, \ldots, T-1\}$.

Let $E$ be the set of all edges that could possibly appear in the process. (For ease of presentation, we take $E = \binom{V}{2}$, however, it will be apparent that our treatment can be extended to graphs with loops, and to directed graphs, with no substantial burden.) For a subset of time-steps $I \subseteq T$, we define the core $\mathcal{G}[I]$ as the maximal subgraph common to all $G_t$ where $t \in I$, i.e., $\mathcal{G}[I] = \{e \in E \mid \forall t \in I : e \in E_t\}$. Our notion of ‘core’ is thus germane to that of a closed subgraph [2, 5, 8]; we prefer the term ‘core’ in order to stress that there is exactly one core in any given time subset $I$. Remarkably, whereas the total number of subgraphs over a given interval may be exponential in the input, the number of cores is bounded by a polynomial [7]. Likewise, while the well-known maximum common subgraph problem is NP-hard in general [4], it becomes polynomially solvable with uniquely labeled vertices.

For any triplet $(t, p, s)$ of integers $t \geq 0$, and $p, s \geq 1$, such that $t + p(s-1) < T$, we set $S(t, p, s) = \{t, t + p, \ldots, t + p(s-1)\}$. We say that $F$ is a periodic subgraph of the dynamic network $\mathcal{G}$ if for some $t, p, s$ we have $F = \mathcal{G}[S(t, p, s)]$ and neither $G_{t-p}$ nor $G_{t+ps}$ contains $F$ as subgraph. If this is the case, we also say that $S(t, p, s)$ is a periodic support set of the periodic subgraph $F$. A periodic subgraph embedding (PSE) is a pair $\langle F, S(t, p, s) \rangle$, where $F = \mathcal{G}[S(t, p, s)]$ is a periodic subgraph and $s \geq \sigma$ for an a-priori set minimum support threshold $\sigma$.

A subgraph $F$ may have several periodic support sets. However, by the maximality conditions, we have that for any two distinct periodic support sets of $F$, say $S(t_1, p_1, s_1)$ and $S(t_2, p_2, s_2)$, (i.e., $\mathcal{G}[S(t_1, p_1, s_1)] = \mathcal{G}[S(t_2, p_2, s_2)]$) such that $p_1 = p_2$, it must hold that: $S(t_1, p_1, s_1) \cap S(t_2, p_2, s_2) = \emptyset$, $t_1 + s_1 p_1 \neq t_2 + s_2 p_2$. 
Let now $P_1 = \langle F, S(t_1, p_1, s_1) \rangle$ and $P_2 = \langle F, S(t_2, p_2, s_2) \rangle$ be two periodic subgraph embeddings of the same periodic subgraph $F$. We say that $P_2$ subsumes $P_1$ iff $S(t_1, p_1, s_1) \not\subseteq S(t_2, p_2, s_2)$. A PSE not subsumed by any other PSE is said to be parsimonious.

Clearly, if $P_2$ subsumes $P_1$ then the following properties hold:

- (i) $p_1 = \lambda p_2$ for some integer $\lambda > 1$;
- (ii) $t_1 \equiv t_2 \pmod{p_2}$;
- (iii) $0 \leq t_1 - t_2 < p_1$ and $0 \leq t_2 + (s_2 - 1)p_2 - (t_1 + (s_1 - 1)p_1) < p_1$.

In words, a PSE $\langle F, S(t, p, s) \rangle$ is not parsimonious if its embedding can be enriched by expanding it into a larger embedding, with a smaller period.

For a graphical example of above definitions, see Figure 1.

\begin{figure}[h]
\centering
\begin{tabular}{cccccccccc}
0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\
\end{tabular}
\caption{Periodic subgraphs, subsumption}
\end{figure}

$G[S(4, 2, 3)] = \{C, D\}$ is a periodic subgraph. $G[1, 3] = \{A, B\}$ is not a periodic subgraph since not temporarily maximal. $G[S(0, 4, 3)]$ and $G[S(4, 2, 3)]$ different periodic subgraphs (different supports).

$G[S(1, 4, 3)] = \{A, B\}$ is a not parsimonious periodic subgraph, because while it is maximal, $G[S(1, 2, 5)]$ subsumes it.

**Problem Statement.** Given a dynamic network $G$ and a minimum support threshold $\sigma \geq 2$, the Periodic Subgraph Mining Problem is to list all parsimonious periodic subgraph embeddings in $G$ that satisfy the minimum support.

**Previous Work.** In [7], Lahiri and Berger-Wolf present a one-pass algorithm which finds all (not necessarily parsimonious) PSEs in time $O((|V| + |E|)T^3 \ln T)$ and space $O((|V| + |E| + p^2_{\text{max}})T^2 \ln T)$, where $p_{\text{max}}$ represents the maximum period and $|E| = \max_t |E_t|$ is the maximum size of an edge set over all time-steps. If all possible periods are considered, then $p_{\text{max}} = \lceil \frac{T-1}{\sigma-1} \rceil$ and the space complexity becomes $O((|V| + |E|)T^2 \ln T + T^4 \sigma \ln T)$. 

3
A modification of this algorithm is also presented in [7] that can filter out non-parsimonious PSEs. An analysis of the time complexity is not supplied for this variant; however, its space complexity is the same as above. A faster, $O((|V| + |E|)T^2 \ln(T/\sigma))$ time implementation of the main algorithm presented in [7] has been proposed recently in [1]. However, this algorithm does not include filtering out non-parsimonious PSEs from the output.

**Main Result.** We present an online algorithm that finds all PSEs of a given dynamic network $\mathcal{G}$ over $T$ time-steps in time $O(|V| + \frac{T^2}{\sigma} |\tilde{E}|)$, where $|\tilde{E}|$ is the average size for an edge set over all time-steps. There can be $\Theta(|V| + \frac{T^2}{\sigma} |\tilde{E}|)$ PSEs in the worst case, and we show that they can be all published within this bound using suitable compact descriptors. Under such a representation, the algorithm is thus optimal.

In addition, we show how to modify our algorithm in order to filter out non-parsimonious PSEs, while in practice keeping the same time complexity.

The remainder of this paper is organized as follows. In the next section we derive the underlying criterion of our approach and provide an algorithm detecting all PSEs. An efficient implementation of the algorithm is discussed in Section 3, and its complexity is analyzed in Section 4. We briefly address subsumption in the last section.

### 2. Finding all PSEs

**The basic idea.** As before, let $\mathcal{G} = \langle G_0, \ldots, G_{T-1} \rangle$ be a dynamic network over a population of vertices $V$. With some abuse of language, any pair $(i, j)$ of vertices with $i < j$ will be called an *edge*. At the generic time $t$ in the evolution of the network, an edge is said to be *active* if it appears in $G_t$, *inactive* otherwise.

For each *period* $p$ ($1 \leq p \leq p_{\text{max}}$) and *phase* $r$ ($0 \leq r < p$), the $(r,p)$-*projection* of $\mathcal{G}$ is the subsequence 

$$\langle G'_0, G'_1, \ldots, G'_{\left\lfloor \frac{T-1-r}{p} \right\rfloor} \rangle$$

defined by $G'_j = G_{r+jp}$, for each $j = 0, 1, \ldots, \left\lfloor (T - r)/p \right\rfloor$. With $\sigma$ denoting the minimum support threshold, the maximum possible period for a PSE is $p_{\text{max}} = \left\lfloor \frac{T-1}{\sigma} \right\rfloor$.

**Observation 1.** Let $\langle F, S(t, p, s) \rangle$ be a PSE for $\mathcal{G}$. Then for $r \equiv t \pmod{p}$ ($0 < r < p$) and $t' = \left\lfloor t/p \right\rfloor$, $\langle F, S(t', 1, s) \rangle$ is a PSE for the $(r,p)$-projection of $\mathcal{G}$.

4
Therefore, we can limit ourselves to the discovery of all PSEs of period 1, and then apply the procedure to the \((r,p)\)-projections of \(G\), for all \(1 < p \leq p_{\text{max}}\) and \(0 \leq r < p\).

**Finding PSEs of period 1 and minimum support \(\sigma\).** We will find PSEs by determining the respective end times of their support. The rationale underlying our approach is captured in the following easy lemma, which states that for finding all PSEs, it is enough to concentrate on those time-steps where some edge becomes inactive after having been active for at least \(\sigma\) consecutive steps.

**Lemma 2.** For each \(t = \sigma - 1, \ldots, T - 1\), there exists a PSE \(\langle F, S(t-s+1,1,s) \rangle\) for some \(s \geq \sigma\) if and only if there is an edge \(e\) such that \(e \in \bigcap_{j=t-\sigma+1}^{t} E_j\) but \(e \notin E_{t+1}\).

**Proof.** Since \(s \geq \sigma\), then every edge in \(F\) is in \(\bigcap_{j=t-\sigma+1}^{t} E_j\). By the maximality of the support, however, there must be at least one edge \(e \in F\) such that \(e \notin E_{t+1}\).

Conversely, assume that there is an edge \(e\) and a value \(s \geq \sigma\) such that \(e \in \bigcap_{j=t-\sigma+1}^{t} E_j\) but \(e \notin E_{t+1}\). We can assume w.l.o.g. that \(s\) is maximum, i.e., \(e \notin E_{t-s}\). Then \(e \in G[S(t-s+1,1,s)]\), whence \(F = G[S(t-s+1,1,s)] \neq \emptyset\), and \(\langle F, S(t-s+1,1,s) \rangle\) is a PSE with support ending at \(t\). \(\square\)

For any given \(t = 0, \ldots, T-1\), let \(L_t\) be the set of edges that at time-step \(t\) inclusive have been active for at least \(\sigma\) consecutive time-steps, and let \(X_t\) be the subset of \(L_t\) consisting of those edges that are inactive in \(E_{t+1}\) (see Figure 2).

Figure 2: The structure of \(L_t\) and \(X_t\)
We will use the criterion of Lemma 2 to set up a procedure that reports all PSEs with support ending at time-step $t$.

For every edge $e \in L_t$, let $\text{last}_t(e)$ be the largest time-step such that $e \in \bigcap_{j=t}^{last_t(e)} E_j$; thus, $e \not\in E_{last_t(e)+1}$. Analogously, let $\text{first}_t(e)$ be the smallest time-step such that $e \in \bigcap_{j=first_t(e)}^{t} E_j$; thus, $e \not\in E_{first_t(e)-1}$. We refer to $\text{first}_t(e)$ and $\text{last}_t(e)$ as the entry time and exit time, respectively, of edge $e$ relative to $t$.

Let $\tau_1^{(t)}, \tau_2^{(t)}, \ldots, \tau_m^{(t)}$ be the distinct entry times for edges in $L_t$, such that $\tau_1^{(t)} < \tau_2^{(t)} < \ldots < \tau_m^{(t)}$, and $B_j^{(t)}$ the set of edges in $L_t$ with entry time $\tau_j^{(t)}$. The procedure in Fig. 3 outputs all PSEs ending at time $t$ for every $t = 1, 2, \ldots, T - 1$.

**Procedure All-PSEs**

1. for every $t = 0, 1, \ldots, T - 1$, s.t. $X_t \neq \emptyset$
2. $F \leftarrow L_t$
3. $j \leftarrow 1$
4. while $F$ contains some edge from $X_t$
5. Output PSE $\langle F, S(\tau_j^{(t)}, 1, t - \tau_j^{(t)} + 1) \rangle$
6. $F \leftarrow F \setminus B_j^{(t)}$
7. $j \leftarrow j + 1$

**Figure 3:** Procedure to output all PSEs ending at time $t$ for every $t = 1, 2, \ldots, T - 1$.

**Lemma 3.** The procedure All-PSEs reports all and only PSEs.

**Proof.** In view of Lemma 2 we only need to prove that the inner part of the for loop correctly reports all and only PSEs with support ending at time $t$.

We first argue that every $\langle F, S(\tau_j^{(t)}, 1, t - \tau_j^{(t)} + 1) \rangle$ (for every $j$) output during the $t$’th iteration of the for loop (lines 2-7) is a PSE ending at $t$. For this, we have to establish the maximality of both $F$ and the support $S(\tau_j^{(t)}, 1, t - \tau_j^{(t)} + 1)$. The maximality of the support follows from the observations: (i) $F \cap X_t \neq \emptyset$, hence $F \not\subseteq E_{t+1}$; (ii) since $F \supseteq B_j^{(t)} \neq \emptyset$, and $B_j^{(t)}$ contains edges with entry time $\tau_j^{(t)}$, then $F \not\subseteq E_{\tau_j^{(t)}-1}$. As for the maximality of $F$, assume that $e \in E_k$ for all $k = \tau_j^{(t)}, \ldots, t$. Thus $\text{first}_t(e) \leq \tau_j^{(t)}$; moreover, since $t - \tau_j^{(t)} + 1 \geq \sigma$, we have that $e \in L_t$. Assume $e \not\in F$. Since
\[ F = L_t \setminus (B_1 \cup B_2 \cup \ldots B_{j-1}) \], there must be a \( j' < j \) such that \( e \in B_{j'} \), implying that \( \text{first}_t(e) = \tau_{j'} > \tau_j \), a contradiction.

For the converse, we have to show that if \( \langle F, S(t', 1, s) \rangle \) is a PSE ending at time-step \( t \), i.e. we have \( t' = t - s + 1 \) and \( s \geq \sigma \), then it is in the output of the procedure ALL-PSEs. For this, we observe that \( F \subseteq L_t \), since \( F \)

must contain only edges which appear in \( E_{t-\sigma+1} \cap \ldots \cap E_t \), and all such edges are in \( L_t \). Also by the definition of a PSE we have that \( F \) must contain an edge \( e \) such that \( e \in E_t \) but \( e \not\in E_{t+1} \). But then \( F \) must contain an edge from \( X_t \). Analogously, there is an edge \( e \) in \( F \) with entering time \( t' \). By the definition of the \( \tau^{(t)} \)s and \( B^{(t)} \)s, this implies that there exists an index \( j \) such that \( \tau^{(t)}_j = t' \). Since all edges in \( F \) have entry time not larger than \( t' \), it follows that \( F \subseteq L_t \setminus \bigcup_{k=1}^{t-1} B^{(t)}_k \). Finally, by the maximality of \( F \) we have that \( F = L_t \setminus \bigcup_{k=1}^{t-1} B^{(t)}_k \), implying that \( \langle F, S(t', 1, s) \rangle \) is indeed output by the procedure during the iteration that corresponds to \( \tau^{(t)}_j = t' \). \( \square \)

3. Implementation

Having fixed a period \( p \) and a phase, we describe first an intuitive paradigm based on a \( \Theta(|V|^2 \times T/p) \) playground, later to be reduced to the \( \Theta(|E| \times T/p) \), where \( |E| \) is the average of the cardinalities of edges in \( E_t \) over all time-steps \( t \). Our playground consists of a \( |V|^2 / 2 \times T / p \) table \( M \) that is used to annotate the evolution of \( G \) as observed over the series of \( T \) snapshots. Specifically, \( M[e, t] = \text{first}_t(e) \) if at time \( t \) the edge \( e \) is active, i.e., belongs to the edge set \( E_t \) of \( G_t \), \( \infty \) otherwise. The table \( M \) features an auxiliary row \( 0 \), which will be used to keep track of which edges become active at a given time-step. It will be maintained in the form of ordered lists.

The columns of \( M \) are filled consecutively one time-step after another, based on the observed graphs. The observation made at time \( t \) will result in an UPDATE and a DETECT, where the action of UPDATE is as follows.

1. Fill up column \( t \): at first initialize all \( M[e, t] \) with \( \infty \) and then for every active edge in \( G_t \) set \( \text{first}_t(e) \) appropriately. That is: propagate the value of \( \text{first}_t(e) \) from \( M[e, t-1] \), if this is a finite value, otherwise set \( \text{first}_t(e) = t \) (at the end \( M[e, t] = \infty \) iff \( e \) is an inactive edge).
2. For any edge \( e \) that becomes active at time \( t \) (i.e., \( M[e, t] = t \) but \( M[e, t-1] = \infty \)), insert \( e \) into the list headed by \( M[0, t] \), which is automatically ordered increasingly. The list is kept as a doubly-linked
list. A pointer is also set up from $M[e, t]$ to the element of $M[0, t]$ which
3. For any edge $e$ that becomes inactive at time $t$ (i.e., $M[e, t] = \infty$ but
   \[M[e, t - 1] \neq \infty\]), do the following:
   (a) if $e$ has less than $\sigma$ active time-steps, then delete $e$ from the ordered
   list $M[0, \text{first}_{t-1}(e)]$;
   (b) otherwise ($((t-1) - \text{first}_{t-1}(e) \geq \sigma$), append the pair $(e, \text{first}_{t-1}(e))$
   to a list $X_{t-1}$.
4. Compute $\tau_{\text{min}}$, the smallest entry time for any element in $X_{t-1}$: for
   example start with $\tau_{\text{min}} = t - 1$, and for any edge $e$ that becomes
   inactive at time $t$, update $\tau_{\text{min}}$ to min($\tau_{\text{min}}$, $\text{first}_{t-1}(e)$).
5. Subdivide the set $L_{t-1}$ in the two sets: $\tilde{L}_{t-1}$, containing the active edges
   with entry time $< \tau_{\text{min}}$ (these edges will not be affected by DETECT, cf.
   Observation 4); and $\hat{L}_{t-1}$ containing the younger edges, to be handled
   at time $t$.
   $\tilde{L}_{t-1}$ is built by scanning column $t - 1$: if $\text{first}_{t-1}(e) < \tau_{\text{min}}$ and $e$
does not become inactive at time $t$, then $e$ is appended to $\tilde{L}_{t-1}$. Note that
since $\tau_{\text{min}} \leq t - \sigma$ we are guaranteed that $e$ has been active for at least
$\sigma$ steps.

All ordering in the previous description refers to the increasing “natural”
order of the edges (as it should be reported when we output a periodic sub-
graph). At this point the control is handed over to DETECT, which must
take care of identifying and publishing those PSEs mandated by $X_{t-1}$. We
begin by making the following observation.

Observation 4. The edges of $L_{t-1}$ with an entry time $\tau_{\text{min}}$ or smaller (these
are $\tilde{L}_{t-1}$ together with the longest living edges exiting at $t - 1$) will be part of
every PSE detected at time-step $t$.

Let $\hat{L}_{t-1}$ be the subset of $L_{t-1}$ consisting of all edges with entry times
$\tau_{\text{min}}$ or bigger. Recall that by definition, the edges in $L_{t-1}$ have been active
at time-step $t - 1$ for at least $\sigma$ steps. Furthermore $X_{t-1} \subseteq \hat{L}_{t-1}$ and the
oldest active edges in $X_{t-1}$ are among the oldest elements in $\hat{L}_{t-1}$. Finally
each PSE ending at time-step $t - 1$ must contain edge(s) from $X_{t-1}$. This
makes it possible that we will use $\hat{L}_{t-1}$ to control the while cycle instead of
$X_{t-1}$.

With reference to the structure of ALL-PSEs, the first order of business
is to build the list $\hat{L}_{t-1}$ and sort its elements by entry time. The naïve
method would be to build at first the list then later sort its elements by standard integer sorting; however, it would charge an undesirable logarithmic overhead.

We will do it in one combined procedure instead, by performing a backward sweep of the 0th row of $\mathbb{M}$, as follows. We scan $\mathbb{M}[0, t - \sigma], \mathbb{M}[0, t - \sigma - 1], \ldots, \mathbb{M}[0, \tau_{\min}]$ and collect all the edges that became active in that time interval, in reverse order of appearance. This yields $\tilde{L}_{t-1}$ in a form of sorted multiple ordered lists, where each sublist is ordered increasingly (by the natural order of the edges), while the sublists are sorted by decreasing entry times.

At the end of this procedure the sets of all active edges (which are active for at least $\sigma$ time-steps) will be represented by the previously built list $\tilde{L}_{t-1}$ (whose edges belong to all PSEs), together with the multi-list $\hat{L}_{t-1}$ (whose edges, besides the longest living ones, do not belong to all PSEs).

With the proviso that in the present implementation of DETECT the PSEs detected at time $t$ are those ending at time $t - 1$, we specify what is involved in the first iteration of the inner loop of ALL-PSEs. DETECT performs Line 5, outputting $F$ (which in the first iteration is $= \tilde{L}_{t-1} \cup \hat{L}_{t-1}$) together with $\tau_{\max}$, the latest entry time of any member of $\hat{L}_{t-1}$. We need now (Line 6) to remove from $F$ all edges having entry time at $\tau_{\max}$ - which is nothing else just update $\hat{L}_{t-1}$ by deleting the first sublist from it. Following that, the same treatment is reapplied to the reduced set, and this is iterated until all elements of $\hat{L}_{t-1}$ have been considered.

This concludes the management of time-step $t$. It is important to recognize that in this way we can successfully control the while cycle simply by checking whether $\hat{L}_{t-1}$ is empty.

Before advancing to time-step $t + 1$, the procedure eliminates from the ordered lists at $\mathbb{M}[0, t - 1], \mathbb{M}[0, t - 2], \ldots, \mathbb{M}[0, \tau_{\min}]$ all first entries of edges that became inactive at time-step $t$. Let $e$ be such an edge. We use the pointer stored during the UPDATE procedure in $\mathbb{M}[e, first_{t-1}(e)]$. This pointer gives the exact position of $e$ in $\mathbb{M}[0, first_{t-1}(e)]$, and the doubly-linked nature of this list allows us to delete this element in constant time. This procedure does not only remove the exited edges from the lists, but it also secures propagation of the important invariant condition, that at any time-step $t$ the cardinality of the union of those lists cannot exceed $|E_t|$.
4. Complexity

We charge some operations to the input (the $\mathcal{M}$-tables, in the current implementation) and some to the output (the collection of PSEs) for period $p$. In particular, the operation of $\text{UPDATE}$ at any given time-step takes time linear in the size of one column of $\mathcal{M}$, whence the work charged by the collection of executions of $\text{UPDATE}$ is $\Theta(|V|^2 \times T/p)$.

Turning to $\text{DETECT}$, we charge the sweep of row 0, which is needed to sort edges by entry time to build $\tilde{L}_{t-1}$, to the entries $\mathcal{M}[0, t-\sigma], \mathcal{M}[0, t-\sigma-1], ..., \mathcal{M}[0, \tau_{\text{min}}]$ of $\mathcal{M}$, that correspond to the segment of one of the longest leaving edge in $X_{t-1}$. Each such segment is charged at most once in this way, while the invariant condition guarantees that the total size of the lists collected in the process is bounded by $|\mathcal{E}_t| < |V|^2/2$, whence the total cost throughout the iterations is $\Theta(|V|^2 \times T/p)$. As is easy to see, the remainder of operations are charged to the output. In particular, every rescanning of the list $L_{t-1}$ mandated by Line 6 of $\text{ALL-PSEs}$ results in one new PSE being published.

Considering now that for each period $p$ we have $p$ projections (one for each phase) of $T/p$ time-steps each, we get that finding all PSEs for period $p$ has a cost of $O(T \times |V|^2)$ plus the time to output the PSEs. There are $T/\sigma$ possible periods. Adding up through all $T/\sigma$ periods yields time $O(T^2 \sigma/\sigma \times |V|^2)$ plus the size of the output, with total space used $\Theta(T \times |V|^2)$.

**List implementation.** It is an easy exercise to modify our implementation in such a way that the role of $\mathcal{M}$ is taken up by a suitable multilist, with rows and columns being played by horizontal and vertical threads, respectively. The resort to a list representation also adds the possibility of monitoring dynamically a varying population, that is, one in which individuals can appear and withdraw, perhaps intermittently, along the way, so that the set $V$ of vertices is not known beforehand. Under this scenario, the order of time yields a straightforward policy to provide the procedure with necessary identifiers for vertices and edges. Specifically, the procedure assigns consecutive integers to vertices and corresponding integer pairs to edges, based on their order of appearance. This makes it possible to expand or contract the record of observations at each consecutive time-step, as needed. We omit the tedious details.

With this implementation $|\mathcal{E}|$ is to be interpreted as the average cardinality of sets of edges $\{E_0, E_1, \ldots, E_{T-1}\}$ over the time-steps of the process. Note that the size of the input is $\Theta(|V| + T \times |\mathcal{E}|)$. Adding up through all
periods yields time $O(|V| + T^2/\sigma \times |\tilde{E}|)$ plus the size of the output, and the total space used is $\Theta(|V| + T \times |\tilde{E}|)$.

**Assessing the output size.** The maximum possible number of PSEs in a dynamic network can be bounded per the following lemma from [7], which is reproduced here together with a short proof.

**Lemma 5 ([7]).** In any dynamic network over $T$ time-steps there are at most $O(T^2 \ln T/\sigma)$ distinct PSEs.

**Proof.** For any fixed period $p$ and time-step $t = 0, 1, \ldots, T - 1$ there can be at most $\lfloor t/p \rfloor$ distinct PSEs with support ending at $t$, namely, one for each possible entry time $t - p, t - 2p, \ldots$. Thus, the total number of PSEs for period $p$ is $O(T^2/p)$. Adding up $T^2/p$ for all periods $p = 1, \ldots, T/\sigma$ in view of the harmonic sum $\sum_{j=1}^{n} 1/j = O(\ln n)$ yields $O(T^2 \ln T/\sigma)$ PSEs. \hfill $\square$

The above bound is tight [7], however, the following lemma yields a better bound for the case of a sparse input.

**Lemma 6.** In any dynamic network over $T$ time-steps there are at most

$$O \left( \frac{T}{\sigma} \sum_{t \in T : X_t \neq \emptyset} |L_t| \right)$$

distinct PSEs.

**Proof.** Fix any fixed period $p$ and time-step $t$, there is at least one PSE ending at $t$ if $X_t \neq \emptyset$, and there are at most $|L_t| \leq |E_t|$ such PSEs. Adding up over all time-steps and over all periods yields the claim. \hfill $\square$

With $|\tilde{E}|$ still denoting the average size of the edge sets in $G$, we can rewrite the above bound as

$$O \left( \frac{T^2}{\sigma} \times |\tilde{E}| \right).$$

Publishing each PSE explicitly at an average cost of $\Theta(|\tilde{E}|)$ time results in total bounds $O(|\tilde{E}| \times T^2 \ln T/\sigma)$ or $O(T^2/\sigma \times |\tilde{E}|^2)$, depending on choice. However, an easy implicit description is possible that encapsulates all PSEs ending at the same time-steps, with the result of eliminating the extra $|\tilde{E}|$ factor in these bounds. For this, observe that for any fixed period $p$ the
subgraphs $F_i$ associated to the PSEs $P_i = \langle F_i, t - s_i + 1, s_i \rangle$ detected at time $t$ form a chain $F_1 \subset F_2 \subset \cdots F_m \subseteq L_t \subseteq E_t$. To identify the whole chain, it is enough to describe the edges of $L_t$ in order of decreasing entry time and the differences $F_i \setminus F_{i-1}$, for $i = 2, \ldots, m$, and to keep track of their entry times. With notations we used in Section 3 we have to report one-by-one the sublists of $\hat{L}_{t-1}$ where the very last sublist is accompanied by the updated $L_{t-1}$.

Under such an implicit description, the overall time complexity reduces thus to

$$O \left( |V| + \frac{T^2}{\sigma} \times |E| \right),$$

which is optimal, in so far as it coincides with the worst-case total size of the output over all periods.

5. Concluding Remarks

We presented an online algorithm to find all periodic subgraphs embeddings (PSEs) of a given dynamic network $G$ over $T$ time-steps that runs faster than previously available methods. In particular, we provide an optimal solution based on an implicit description of the output subgraphs.

It seems difficult to extend the proposed paradigm to filter out non-parsimonious PSEs without forfeiting the online feature. Recall that a PSE $P = \langle F, S(t, p, s) \rangle$ is subsumed by another PSE $P' = \langle F', S(t', p', s') \rangle$ if $F = F'$ ($= G[S(t, p, s)] = G[S(t', p', s')]$) and $S(t, p, s) \subset S(t', p', s')$. One can think of at least two approaches to detecting this phenomenon. The fist one may be called a posteriori filtering: when we detect a PSE $P = \langle F, S(t, p, s) \rangle$, we check whether the same core $F$ forms a PSE for a period $p'$ where $p'$ is a divisor of $p$.

The other one can be called a priori filtering: we store each observed PSE, irrespectively of whether it was reported or already found to be subsumed, and we compare any newly discovered PSE with this “database”.

Here we detail a posteriori filtering and leave the rest for an exercise.

Consider the PSEs in order of increasing period. Faced with the PSE $P = \langle F, S(t, p, s) \rangle$, check, for each $p'$ which divides $p$, and for each $t' \equiv t \pmod {p'}$, $t - p < t' \leq t$, whether there is a PSE $P' = \langle F, S(t', p', s') \rangle$ with $t' + p'(s' - 1) \geq t + p(s - 1)$.

The following observation reduces checking only to a subset of the possible values of $t'$ and $p'$.
Observation 7. Suppose that $PSE \langle F, S(t, p, s) \rangle$ is subsumed by some other $PSE$. Then there exists a prime number $\lambda$ such that $\lambda$ divides $p$ and for each $e \in F$ we have that $e \in \bigcap_{k=0}^{k_{\lambda}} E_{t+p/\lambda}^{k}$ for $p' = p/\lambda$.

If the lists $L_i$ for each projection are made available, then the condition on $e$ in the statement of Observation 7 is verified by checking whether for $r \equiv t \pmod{p'}$ ($0 \leq r < p'$) and $i = \left\lfloor \frac{t}{p'} \right\rfloor$ in the $(r, p')$-projection, it is $last_i(e) \geq t + (s - 1)p$ in the list $L_i$.

Since we are considering periods in order of increasing value, then when handling $p$ we have already computed, in particular, the PSEs for every period $p' = p/\lambda$, where $\lambda$ is a prime divisor of $p$. Then, we can check whether a PSE $\langle F, S(t, p, s) \rangle$ is subsumed by considering only the periods $p'$ given by these $\lambda$’s.

Following standard notation, let $\omega(p)$ denote the number of distinct prime divisors of $p$. Then we can check whether $\langle F, S(t, p, s) \rangle$ is subsumed by some other PSE in time $O(|F|\omega(p))$. This term is to be added to the time needed for producing the output for the PSE $P$.

Acknowledgements. The authors are indebted to F. Cicalese for many helpful discussions. This research was carried out in part while the authors worked together at the Alfréd Rényi Institute in Budapest, with support from the Hungarian Bioinformatics MTKD-CT-2006-042794, Marie Curie Host Fellowships for Transfer of Knowledge. Additional support for Alberto Apostolico was provided by the United States-Israel Binational Science Foundation (BSF) Grant No. 2008217, and by the Research Program of Georgia Tech. Péter L. Erdős was in part supported by the Alexander von Humboldt Foundation and the Hungarian NSF, under contracts NK 78439 and K 68262. Ervin Győri was in part supported by the Hungarian NSF, under contract NK 78439. Cinzia Pizzi was in part supported by Progetto Cariparo: 2008-2011 Systems Biology Approaches to Infer Gene and Protein Time-series Expression Data, and Progetto Ateneo: 2008-2010 Computational Assessment of Protein-Protein Interaction Networks and Validation as a Guide for Modern System Biology.

References


