Journal of Discrete Algorithms 17 (2012) 24–30

Contents lists available at SciVerse ScienceDirect

Journal of Discrete Algorithms

www.elsevier.com/locate/jda

Efficient algorithms for the periodic subgraphs mining problem

Alberto Apostolico [a], Péter L. Erdős [c], Ervin Győri [c], Zsuzsanna Lipták [d], Cinzia Pizzi [b,*]

[a] College of Computing, Georgia Institute of Technology, Atlanta, GA, USA
[b] Dipartimento di Ingegneria dell'Informazione, Università di Padova, Italy
[c] Alfréd Rényi Institute of Mathematics, Budapest, Hungary
[d] Dipartimento di Informatica, Università di Verona, Italy

ARTICLE INFO

Article history:
Available online 23 May 2012

Keywords:
Discovery algorithms
Sequences of networks
Social networks
Periodicity
Interactions

ABSTRACT

Given a sequence $\mathcal{G} = \langle G_0, \ldots, G_{T-1} \rangle$ of simple graphs over uniquely labeled vertices from a set V, the periodic subgraph mining problem consists in discovering maximal subgraphs that recur at regular intervals in $\mathcal{G}$. 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 in any direction. We give algorithms that improve the theoretical complexity of solutions 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 $\mathcal{G}$—to publish all maximal periodic subgraphs that meet or exceed a minimum occurrence threshold σ.

1. Preliminaries

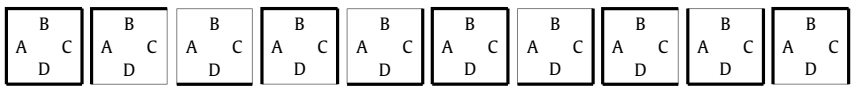
In recent years, graph theoretical methods have found increasing use in the social sciences, where individuals or entities which interact in a dynamic *social* network may be represented by the vertices of a graph, and an interaction between any pair of individuals is represented by an edge. The evolution 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, including the periodically recurrent subgraphs which are the subject of the present paper. Apart from obvious contexts such as social networks [10] and internet topology analysis [4], problems of this nature have emerged, for example, in the study of mating patterns within communities of wild animals [7], in inferring predictable behaviors from sensor networks [3], and in the study of usage of mobile devices.

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* $\mathbb{T} = \{0, 1, \ldots, T-1\}$.

Let $\mathbb{E}$ be the set of all edges that could possibly appear in the process. For ease of presentation, we take $\mathbb{E} = \binom{V}{2}$, 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 $I \subseteq \mathbb{T}$, we define the *core* $\mathcal{G}[I]$ as the maximal subgraph common to all G_t where $t \in I$, that is, $\mathcal{G}[I] = \{e \in \mathbb{E} \mid \forall t \in I\colon\ e \in E_t\}$. Our notion of 'core' is thus germane to that of a *closed subgraph* [2,6,9]; 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

* Corresponding author.
E-mail addresses: axa@cc.gatech.edu (A. Apostolico), elp@renyi.hu (P.L. Erdős), ervin@renyi.hu (E. Győri), zsuzsanna.liptak@univr.it (Zs. Lipták), cinzia.pizzi@dei.unipd.it (C. Pizzi).

1570-8667/$ – see front matter
http://dx.doi.org/10.1016/j.jda.2012.05.002



$\mathcal{G}[S(4,2,3)] = \{C, D\}$ is a **periodic subgraph**.
$\langle \{A, B\}, S(1,2,2)\rangle = \langle \{A, B\}, \{1,3\}\rangle$ is not a periodic subgraph embedding because its **support set is not maximal**.
$\langle \{C, D\}, S(0,4,3)\rangle$ and $\langle \{C, D\}, S(4,2,3)\rangle$ are two **different PSEs**, even though they both have the same core $\{C, D\}$.
The PSE $\langle \{A, B\}, S(1,4,3)\rangle$ is **not parsimonious**, because while it is maximal, the PSE $\langle \{A, B\}, S(1,2,5)\rangle$ **subsumes** it.

Fig. 1. Periodic subgraphs, PSEs, subsumption.

polynomial [8]. Likewise, while the well-known *maximum common subgraph* problem is NP-hard in general [5], it becomes solvable in polynomial time with uniquely labeled vertices.

For any triplet (t, p, s) of integers $t \geqslant 0$, and $p, s \geqslant 1$, 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 $s \geqslant \sigma$ for an a-priori set *minimum support threshold* σ.

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 $S(t_2,p_2,s_2)$ (that is, $\mathcal{G}[S(t_1,p_1,s_1)] = \mathcal{G}[S(t_2,p_2,s_2)]$), such that $p_1 = p_2$, it must hold that: $S(t_1,p_1,s_1) \cap S(t_2,p_2,s_2) = \emptyset$, $t_1 + s_1 p_1 \neq t_2$, and $t_2 + s_2 p_2 \neq t_1$.

Let $P_1 = \langle F, S(t_1,p_1,s_1)\rangle$ 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 if and only if $S(t_1,p_1,s_1) \subsetneq S(t_2,p_2,s_2)$. 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) $0 \leqslant t_1 - t_2 < p_1$ and $0 \leqslant t_2 + (s_2 - 1)p_2 - (t_1 + (s_1 - 1)p_1) < p_1$.

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 the above definitions, see Fig. 1.

1.1. Problem statement

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

1.2. Previous work

In [8], 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 $O((|V|+|E|+p_{\max}^2)T^2 \ln T)$, where $p_{\max}$ 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 $p_{\max} = \lfloor \frac{T-1}{\sigma-1} \rfloor$ and the space complexity becomes $O((|V|+|E|)T^2 \ln T + \frac{T^4}{\sigma^2} \ln T)$.

A modification of this algorithm is also presented in [8] 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 better, $O((|V|+|E|)T^2 \ln(T/\sigma))$ time implementation of the main algorithm presented in [8] has been proposed recently in [1]. However, this algorithm does not include filtering out non-parsimonious PSEs from the output.

1.3. 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.

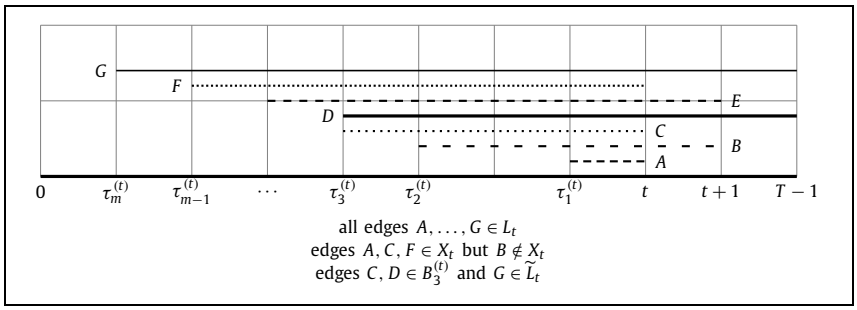


Fig. 2. The structure of L_t and X_t.

In addition, we sketch 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

2.1. 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, and is *inactive* otherwise.

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

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

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

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 $t' = \lfloor t/p \rfloor$, $\langle F, S(t', 1, s) \rangle$ is a PSE for the (r, p)-projection of $\mathcal{G}$.*

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 $\mathcal{G}$, for all $1 < p \leqslant p_{\max}$ and $0 \leqslant r < p$.

2.2. Finding PSEs of period 1 and minimum support σ

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

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 σ 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 Fig. 2).

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 $last_t(e)$ be the largest time-step such that $e \in \bigcap_{j=t}^{last_t(e)} E_j$; thus, $e \notin E_{last_t(e)+1}$. Analogously, let $first_t(e)$ be the smallest time-step such that $e \in \bigcap_{j=first_t(e)}^{t} E_j$; thus, $e \notin E_{first_t(e)-1}$. We refer to $first_t(e)$ and $last_t(e)$ as the *entry time* and *exit time*, respectively, of edge e *relative to* t.

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

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

Let $\tau_1^{(t)}, \tau_2^{(t)}, \ldots, \tau_m^{(t)}$ be the distinct entry times for edges in L_t, such that $\tau_m^{(t)} < \tau_{m-1}^{(t)} < \cdots < \tau_1^{(t)}$, 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 $t = 1, 2, \ldots, T-1$.

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 tth 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 $\mathit{first}_t(e) \leqslant \tau_j^{(t)}$; moreover, since $t - \tau_j^{(t)} + 1 \geqslant \sigma$, we have that $e \in L_t$. Assume $e \notin F$. 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 $\mathit{first}_t(e) = \tau_{j'} > \tau_j$, 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, that is we have $t' = t - s + 1$ and $s \geqslant \sigma$, 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 $E_{t-\sigma+1} \cap \cdots \cap E_t$, 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 $e \notin E_{t+1}$. 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 $\tau_j^{(t)} = t'$. Since all edges in F have entry time not larger than t', it follows that $F \subseteq L_t \setminus \bigcup_{k=1}^{j-1} B_k^{(t)}$. Finally, by the maximality of F we have that $F = L_t \setminus \bigcup_{k=1}^{j-1} B_k^{(t)}$, implying that $\langle F, S(t', 1, s)\rangle$ is indeed output by the procedure during the iteration that corresponds to $\tau_j^{(t)} = t'$. □

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 $\Theta(|\tilde{E}| \times T/p)$, where $|\tilde{E}|$ 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 $\mathbb{M}$ that is used to annotate the evolution of $\mathcal{G}$ as observed over the series of T snapshots. Specifically, $\mathbb{M}[e, t] = \mathit{first}_t(e)$ if at time t the edge e is active, that is, belongs to the edge set E_t of G_t, ∞ otherwise. The table $\mathbb{M}$ 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 $\mathbb{M}$ 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 $\mathbb{M}[e, t]$ with ∞ and then for every active edge in G_t set $\mathit{first}_t(e)$ appropriately. That is: propagate the value of $\mathit{first}_t(e)$ from $\mathbb{M}[e, t-1]$, if this is a finite value, otherwise set $\mathit{first}_t(e) = t$ (at the end $\mathbb{M}[e, t] = \infty$ if and only if e is an inactive edge).
2. For any edge e that becomes active at time t (that is, $\mathbb{M}[e, t] = t$ but $\mathbb{M}[e, t-1] = \infty$), insert e into the list headed by $\mathbb{M}[0, t]$, which is automatically ordered increasingly. The list is kept as a doubly-linked list. A pointer is also set up from $\mathbb{M}[e, t]$ to the element of $\mathbb{M}[0, t]$ which corresponds to e.
3. For any edge e that becomes inactive at time t (that is, $\mathbb{M}[e, t] = \infty$ but $\mathbb{M}[e, t-1] \neq \infty$), do the following:
   (a) if e has less than σ active time-steps, then delete e from the ordered list $\mathbb{M}[0, \mathit{first}_{t-1}(e)]$;
   (b) otherwise ($(t-1) - \mathit{first}_{t-1}(e) \geqslant \sigma$), append the pair $(e, \mathit{first}_{t-1}(e))$ to a list X_{t-1}.
4. Compute τ_{min}, the smallest entry time for any element in X_{t-1}: for example start with $\tau_{min} = t - 1$, and for any edge e that becomes inactive at time t, update τ_{min} to $\min(\tau_{min}, \mathit{first}_{t-1}(e))$.
5. Subdivide the set L_{t-1} in the two sets: $\widetilde{L}_{t-1}$, containing the active edges with entry time $< \tau_{min}$ (these edges will not be affected by DETECT, cf. Observation 4); and $\widehat{L}_{t-1}$ containing the younger edges, to be handled at time t.

$\widetilde{L}_{t-1}$ is built by scanning column $t-1$: if $first_{t-1}(e) < \tau_{min}$ and e does not become inactive at time t, then e is appended to $\widetilde{L}_{t-1}$. Note that since $\tau_{min} \leqslant t-\sigma$ we are guaranteed that e has been active for at least σ 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 subgraph). 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 τ_{min} or smaller (these are $\widetilde{L}_{t-1}$ together with the longest living edges exiting at $t-1$) will be part of every PSE detected at time-step t.*

Let $\widehat{L}_{t-1}$ be the subset of L_{t-1} consisting of all edges with entry times τ_{min} or bigger. Recall that by definition, the edges in L_{t-1} have been active at time-step $t-1$ for at least σ steps. Furthermore $X_{t-1} \subseteq \widehat{L}_{t-1}$ and the oldest active edges in X_{t-1} are among the oldest elements in $\widehat{L}_{t-1}$. 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 $\widehat{L}_{t-1}$ 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 $\widehat{L}_{t-1}$ 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 $\widehat{L}_{t-1}$ 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 σ time-steps) will be represented by the previously built list $\widetilde{L}_{t-1}$ (whose edges belong to all PSEs), together with the multi-list $\widehat{L}_{t-1}$ (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 $= \widetilde{L}_{t-1} \cup \widehat{L}_{t-1}$) together with τ_{max}, the latest entry time of any member of $\widehat{L}_{t-1}$. We need now (Line 6) to remove from F all edges having entry time at τ_{max}—which only requires us to delete the first sublist from $\widehat{L}_{t-1}$. Following that, the same treatment is reapplied to the reduced set, and this is iterated until all elements of $\widehat{L}_{t-1}$ 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 $\widehat{L}_{t-1}$ 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 and 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 $\mathbb{M}$-tables, in the current implementation) and some to the output (the collection of PSEs) for period p. In particular, the operation of UPDATE at any given time-step takes time linear in the size of one column of $\mathbb{M}$, whence the work charged by the collection of executions of UPDATE is $\Theta(|V|^2 \times T/p)$.

Turning to DETECT, we charge the sweep of row 0, which is needed to sort edges by entry time to build $\widehat{L}_{t-1}$, to the entries $\mathbb{M}[0, t-\sigma]$, $\mathbb{M}[0, t-\sigma-1], \ldots, \mathbb{M}[0, \tau_{min}]$ of $\mathbb{M}$, 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 $|E_t| < |V|^2/2$, whence the total cost throughout the iterations is $\Theta(|V|^2 \times T/p)$. The remainder of operations are charged to the output. In particular, every rescanning of the list L_{t-1} mandated by Line 6 of 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/σ possible periods. Adding up through all T/σ periods yields time $O(\frac{T^2}{\sigma} \times |V|^2)$ plus the size of the output, with total space used $\Theta(T \times |V|^2)$.

4.1. List implementation

It is an easy exercise to modify our implementation in such a way that the role of $\mathbb{M}$ is taken up by a suitable multi-list, with rows and columns being played by horizontal and vertical threads, respectively. The resort to a list representation also adds the possibility of dynamically monitoring 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 $|\tilde{E}|$ 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 $\Theta(|V| + T \times |\tilde{E}|)$. 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 $\Theta(|V| + T \times |\tilde{E}|)$.

4.2. Assessing the output size

The maximum possible number of PSEs in a dynamic network can be bounded per the following lemma from [8], which is reproduced here together with a short proof.

Lemma 5. *(See [8].) 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. □

The above bound is tight [8], 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*

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

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 $|L_t| \leqslant |E_t|$ such PSEs. Adding up over all time-steps and over all periods yields the claim. □

With $|\tilde{E}|$ still denoting the average size of the edge sets in $\mathcal{G}$, 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 $O(\frac{T^2}{\sigma} \times |\tilde{E}|^2)$, 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. To see 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 $\widehat{L}_{t-1}$ 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

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

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 $\mathcal{G}$ over T time-steps with an improved time complexity with respect to 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 $F = F'$ $(= \mathcal{G}[S(t, p, s)] = \mathcal{G}[S(t', p', s')])$ and $S(t, p, s) \subset S(t', p', s')$. One can think of at least two approaches to detecting this phenomenon. The first 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 and has already been computed.

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 $t' \equiv t \pmod{p'}$, $t - p < t' \leqslant t$, whether there is a PSE $P' = \langle F, S(t', p', s')\rangle$ with $t' + p'(s' - 1) \geqslant t + p(s - 1)$.

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 λ such that λ divides p and for each $e \in F$ we have that $e \in \bigcap_{k=0}^{s\times\lambda} E_{t+p'k}$ 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 $r \equiv t \pmod{p'}$ $(0 \leqslant r < p')$ and $i = \lfloor \frac{t}{p'} \rfloor$ in the (r, p')-projection, it is $last_i(e) \geqslant t + (s - 1)p$ in the list L_i.

Since we are considering periods in order of increasing value, when handling p we have already computed the PSEs for every period $p' = p/\lambda$, where λ 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 λ'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 to produce the output for the PSE P.

5.1. Future directions

An interesting evolution of our work would be to deal with approximate time periodicity, in which subgraphs can actually occur at time distance $p \pm \delta$, for a fixed δ. We are currently exploring ways to deal with this problem which clearly would play an important role in real life applications.

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.

We thank an anonymous referee for carefully reading the manuscript and making numerous useful suggestions.

References

[1] A. Apostolico, M. Barbares, C. Pizzi, Speedup for a periodic subgraph miner, Information Processing Letters 111 (2011) 521–523.
[2] C. Carpineto, G. Romano, Concept Data Analysis: Theory and Applications, John Wiley & Sons Inc., Chichester, England, 2004.
[3] N. Eagle, A. (Sandy) Pentland, Reality mining: sensing complex social systems, Personal Ubiquitous Computing 10 (2006) 255–268.
[4] M. Faloutsos, P. Faloutsos, C. Faloutsos, On power-law relationships of the internet topology, in: Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication, ACM Press, 1999, pp. 251–262.
[5] M.R. Garey, D.S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, W.H. Freeman & Co., New York, NY, USA, 1979.
[6] J. Han, H. Cheng, D. Xin, X. Yan, Frequent pattern mining: current status and future directions, Data Mining and Knowledge Discovery 14 (2007) 55–86.
[7] P. Juang, H. Oki, Y. Wang, M. Martonosi, L.S. Peh, D. Rubenstein, Energy-efficient computing for wildlife tracking: design tradeoffs and early experiences with zebranet, in: ASPLOS-X: Proceedings of the 10th International Conference on Architectural Support for Programming Languages and Operating Systems, ACM Press, New York, NY, USA, 2002, pp. 96–107.
[8] M. Lahiri, T. Berger-Wolf, Periodic subgraph mining in dynamic networks, Knowledge and Information Systems 24 (2010) 467–497.
[9] N. Pasquier, Y. Bastide, R. Taouil, L. Lakhal, Efficient mining of association rules using closed itemset lattices, Information Systems 24 (1999) 25–46.
[10] S. Wasserman, K. Faust, Social Network Analysis: Methods and Applications, Cambridge University Press, Cambridge, England, 1994.