

# Efficient Algorithms for the Periodic Subgraphs Mining Problem

Alberto Apostolico<sup>a,b</sup>, Péter L. Erdős<sup>c</sup>, Ervin Gyóri<sup>c</sup>, Zsuzsanna Lipták<sup>d</sup>,  
Cinzia Pizzi<sup>b</sup>

<sup>a</sup>*College of Computing, Georgia Institute of Technology, Atlanta, USA*

<sup>b</sup>*Dipartimento di Ingegneria dell'Informazione, University of Padova, Italy*

<sup>c</sup>*Alfréd Rényi Institute of Mathematics, Budapest, Hungary*

<sup>d</sup>*AG Genominformatik, Technische Fakultät, Bielefeld University, Germany*

---

## Abstract

Given a sequence  $\mathcal{G} = \langle G_0, \dots, G_{T-1} \rangle$  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  $\mathcal{G}$ . 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 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  $\mathcal{G}$ —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

2 In recent years, graph theoretical methods have found increasing use in  
3 the social sciences, where individuals or entities interacting pairwise in a dy-  
4 namic *social* network may be represented by the vertices of a graph, and

---

*Email addresses:* [axa@cc.gatech.edu](mailto:axa@cc.gatech.edu) (Alberto Apostolico), [elp@renyi.hu](mailto:elp@renyi.hu) (Péter L. Erdős), [ervin@renyi.hu](mailto:ervin@renyi.hu) (Ervin Gyóri), [zsuzsa@cebitec.uni-bielefeld.de](mailto:zsuzsa@cebitec.uni-bielefeld.de) (Zsuzsanna Lipták), [cinzia.pizzi@dei.unipd.it](mailto:cinzia.pizzi@dei.unipd.it) (Cinzia Pizzi)

5 an interaction between any pair of individuals is represented by an edge.  
6 The evolution of such a network is then modeled as a sequence of graphs,  
7 in which edges over a set of uniquely labeled vertices evolve while a series  
8 of observations is made at regular intervals. In this context, it has become  
9 of interest to detect the emergence of various regularities, among which the  
10 periodically recurrent subgraphs that form the subject of the present paper.  
11 Apart from obvious contexts such as web communities, problems of this nature  
12 have emerged, e.g. in the study of mating patterns within communities  
13 of wild animals [6], in inferring predictable behaviors from sensor networks  
14 [3], etc.

15 Formally, by a *dynamic network* we understand a sequence  $\mathcal{G} = \langle G_0, \dots,$   
16  $G_{T-1} \rangle$  of simple graphs  $G_t = (V_t, E_t)$ , where each  $V_t$  is a subset of the same,  
17 uniquely labeled vertex set  $V$ . The “dynamics” of the network is captured by  
18 the ever-changing edge sets  $E_t$  over a series of *time-steps*  $\mathbb{T} = \{0, 1, \dots, T-1\}$ .

19 Let  $\mathbb{E}$  be the set of all edges that could possibly appear in the process.  
20 (For ease of presentation, we take  $\mathbb{E} = \binom{V}{2}$ , however, it will be apparent that  
21 our treatment can be extended to graphs with loops, and to directed graphs,  
22 with no substantial burden.) For a subset of time-steps  $I \subseteq \mathbb{T}$ , we define  
23 the *core*  $\mathcal{G}[I]$  as the maximal subgraph common to all  $G_t$  where  $t \in I$ , i.e.,  
24  $\mathcal{G}[I] = \{e \in \mathbb{E} \mid \forall t \in I : e \in E_t\}$ . Our notion of ‘core’ is thus germane to that  
25 of a *closed subgraph* [2, 5, 8]; we prefer the term ‘core’ in order to stress that  
26 there is exactly one core in any given time subset  $I$ . Remarkably, whereas the  
27 total number of subgraphs over a given interval may be exponential in the  
28 input, the number of cores is bounded by a polynomial [7]. Likewise, while  
29 the well-known *maximum common subgraph* problem is NP-hard in general  
30 [4], it becomes polynomially solvable with uniquely labeled vertices.

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

39 A subgraph  $F$  may have several periodic support sets. However, by the  
40 maximality conditions, we have that for any two distinct periodic support sets  
41 of  $F$ , say  $S(t_1, p_1, s_1)$  and  $S(t_2, p_2, s_2)$ , (i.e.,  $\mathcal{G}[S(t_1, p_1, s_1)] = \mathcal{G}[S(t_2, p_2, s_2)]$ )  
42 such that  $p_1 = p_2$ , it must hold that:  $S(t_1, p_1, s_1) \cap S(t_2, p_2, s_2) = \emptyset, t_1 + s_1 p_1 \neq$

43  $t_2$ , and  $t_2 + s_2 p_2 \neq t_1$ .

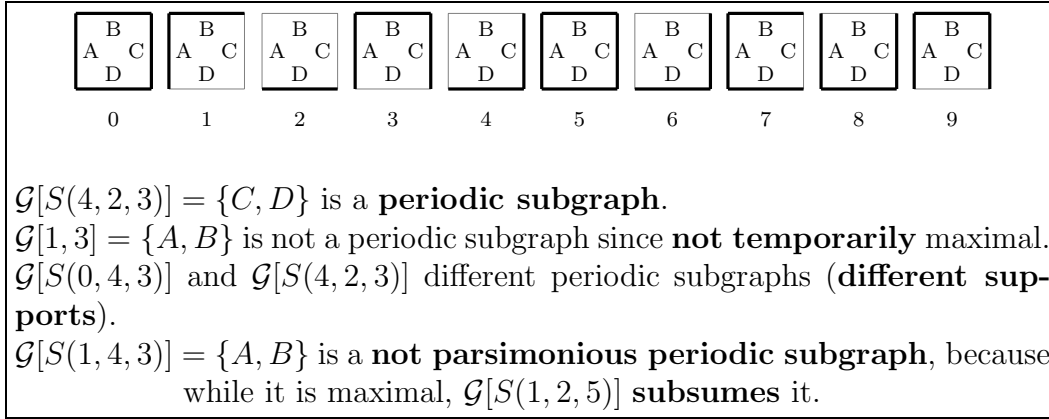
44 Let now  $P_1 = \langle F, S(t_1, p_1, s_1) \rangle$  and  $P_2 = \langle F, S(t_2, p_2, s_2) \rangle$  be two periodic  
 45 subgraph embeddings of the same periodic subgraph  $F$ . We say that  $P_2$  *sub-*  
 46 *sumes*  $P_1$  iff  $S(t_1, p_1, s_1) \subsetneq S(t_2, p_2, s_2)$ . A PSE not subsumed by any other  
 47 PSE is said to be *parsimonious*.

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

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

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

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



**Figure 1:** Periodic subgraphs, subsumption

55 **Problem Statement.** *Given a dynamic network  $\mathcal{G}$  and a minimum support*  
 56 *threshold  $\sigma \geq 2$ , the Periodic Subgraph Mining Problem is to list all parsimo-*  
 57 *nious periodic subgraph embeddings in  $\mathcal{G}$  that satisfy the minimum support.*

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

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

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

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

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

## 83 2. Finding all PSEs

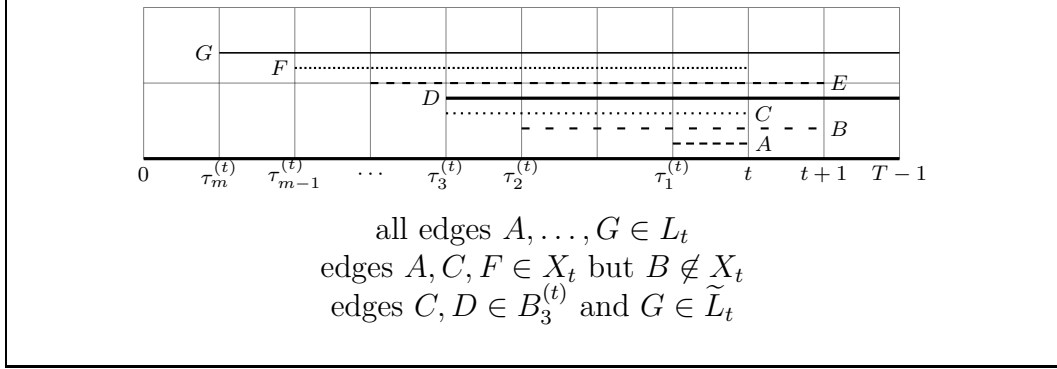
84 **The basic idea.** As before, let  $\mathcal{G} = \langle G_0, \dots, G_{T-1} \rangle$  be a dynamic network  
 85 over a population of vertices  $V$ . With some abuse of language, any pair  $(i, j)$   
 86 of vertices with  $i < j$  will be called an *edge*. At the generic time  $t$  in the  
 87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

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

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

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

92 **Observation 1.** Let  $\langle F, S(t, p, s) \rangle$  be a PSE for  $\mathcal{G}$ . Then for  $r \equiv t \pmod{p}$   
 93 ( $0 < r < p$ ) and  $t' = \lfloor t/p \rfloor$ ,  $\langle F, S(t', 1, s) \rangle$  is a PSE for the  $(r, p)$ -  
 94 *projection* of  $\mathcal{G}$ .



**Figure 2:** The structure of  $L_t$  and  $X_t$

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

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

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

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

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

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

118 We will use the criterion of Lemma 2 to set up a procedure that reports  
 119 all PSEs with support ending at time-step  $t$ .

120 For every edge  $e \in L_t$ , let  $last_t(e)$  be the largest time-step such that  $e \in$   
 121  $\bigcap_{j=t}^{last_t(e)} E_j$ ; thus,  $e \notin E_{last_t(e)+1}$ . Analogously, let  $first_t(e)$  be the smallest  
 122 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  
 123  $first_t(e)$  and  $last_t(e)$  as the *entry time* and *exit time*, respectively, of edge  $e$   
 124 *relative to  $t$* .

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

**Procedure ALL-PSES**

1. **for** every  $t = 0, 1, \dots, T - 1$ , 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  $t = 1, 2, \dots, T - 1$ .

129 **Lemma 3.** *The procedure ALL-PSES reports all and only PSEs.*

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

132 We first argue that every  $\langle F, S(\tau_j^{(t)}, 1, t - \tau_j^{(t)} + 1) \rangle$  (for every  $j$ ) output  
 133 during the  $t$ 'th iteration of the **for** loop (lines 2-7) is a PSE ending at  $t$ .  
 134 For this, we have to establish the maximality of both  $F$  and the support  
 135  $S(\tau_j^{(t)}, 1, t - \tau_j^{(t)} + 1)$ . The maximality of the support follows from the obser-  
 136 vations: (i)  $F \cap X_t \neq \emptyset$ , hence  $F \not\subseteq E_{t+1}$ ; (ii) since  $F \supseteq B_j^{(t)} \neq \emptyset$ , and  $B_j^{(t)}$   
 137 contains edges with entry time  $\tau_j^{(t)}$ , then  $F \not\subseteq E_{\tau_j^{(t)}-1}$ . As for the maximal-  
 138 ity of  $F$ , assume that  $e \in E_k$  for all  $k = \tau_j^{(t)}, \dots, t$ . Thus  $first_t(e) \leq \tau_j^{(t)}$ ;  
 139 moreover, since  $t - \tau_j^{(t)} + 1 \geq \sigma$ , we have that  $e \in L_t$ . Assume  $e \notin F$ . Since

140  $F = L_t \setminus (B_1 \cup B_2 \cup \dots B_{j-1})$ , there must be a  $j' < j$  such that  $e \in B_{j'}$ ,  
 141 implying that  $first_t(e) = \tau_{j'} > \tau_j$ , a contradiction.

142 For the converse, we have to show that if  $\langle F, S(t', 1, s) \rangle$  is a PSE ending  
 143 at time-step  $t$ , i.e. we have  $t' = t - s + 1$  and  $s \geq \sigma$ , then it is in the output  
 144 of the procedure ALL-PSES. For this, we observe that  $F \subseteq L_t$ , since  $F$   
 145 must contain only edges which appear in  $E_{t-\sigma+1} \cap \dots \cap E_t$ , and all such edges  
 146 are in  $L_t$ . Also by the definition of a PSE we have that  $F$  must contain an  
 147 edge  $e$  such that  $e \in E_t$  but  $e \notin E_{t+1}$ . But then  $F$  must contain an edge  
 148 from  $X_t$ . Analogously, there is an edge  $e$  in  $F$  with entering time  $t'$ . By the  
 149 definition of the  $\tau^{(t)}$ s and  $B^{(t)}$ s, this implies that there exists an index  $j$  such  
 150 that  $\tau_j^{(t)} = t'$ . Since all edges in  $F$  have entry time not larger than  $t'$ , it  
 151 follows that  $F \subseteq L_t \setminus \bigcup_{k=1}^{j-1} B_k^{(t)}$ . Finally, by the maximality of  $F$  we have  
 152 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  
 153 procedure during the iteration that corresponds to  $\tau_j^{(t)} = t'$ .  $\square$

### 154 3. Implementation

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

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

- 167 1. Fill up column  $t$ : at first initialize all  $\mathbb{M}[e, t]$  with  $\infty$  and then for every  
 168 active edge in  $G_t$  set  $first_t(e)$  appropriately. That is: propagate the  
 169 value of  $first_t(e)$  from  $\mathbb{M}[e, t-1]$ , if this is a finite value, otherwise set  
 170  $first_t(e) = t$  (at the end  $\mathbb{M}[e, t] = \infty$  iff  $e$  is an inactive edge).
- 171 2. For any edge  $e$  that becomes active at time  $t$  (i.e.,  $\mathbb{M}[e, t] = t$  but  
 172  $\mathbb{M}[e, t-1] = \infty$ ), insert  $e$  into the list headed by  $\mathbb{M}[0, t]$ , which is  
 173 automatically ordered increasingly. The list is kept as a doubly-linked

- 174 list. A pointer is also set up from  $\mathbb{M}[e, t]$  to the element of  $\mathbb{M}[0, t]$  which  
175 corresponds to  $e$ .
- 176 3. For any edge  $e$  that becomes inactive at time  $t$  (i.e.,  $\mathbb{M}[e, t] = \infty$  but  
177  $\mathbb{M}[e, t - 1] \neq \infty$ ), do the following:
- 178 (a) if  $e$  has less than  $\sigma$  active time-steps, then delete  $e$  from the ordered  
179 list  $\mathbb{M}[0, first_{t-1}(e)]$  ;
- 180 (b) otherwise ( $(t-1) - first_{t-1}(e) \geq \sigma$ ), append the pair  $(e, first_{t-1}(e))$   
181 to a list  $X_{t-1}$ .
- 182 4. Compute  $\tau_{min}$ , the smallest entry time for any element in  $X_{t-1}$ : for  
183 example start with  $\tau_{min} = t - 1$ , and for any edge  $e$  that becomes  
184 inactive at time  $t$ , update  $\tau_{min}$  to  $\min(\tau_{min}, first_{t-1}(e))$ .
- 185 5. Subdivide the set  $L_{t-1}$  in the two sets:  $\tilde{L}_{t-1}$ , containing the active edges  
186 with entry time  $< \tau_{min}$  (these edges will not be affected by DETECT, cf.  
187 Observation 4); and  $\hat{L}_{t-1}$  containing the younger edges, to be handled  
188 at time  $t$ .
- 189  $\tilde{L}_{t-1}$  is built by scanning column  $t - 1$ : if  $first_{t-1}(e) < \tau_{min}$  and  $e$  does  
190 not become inactive at time  $t$ , then  $e$  is appended to  $\tilde{L}_{t-1}$ . Note that  
191 since  $\tau_{min} \leq t - \sigma$  we are guaranteed that  $e$  has been active for at least  
192  $\sigma$  steps.

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

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

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

208 With reference to the structure of ALL-PSEs, the first order of business  
209 is to build the list  $\hat{L}_{t-1}$  and sort its elements by entry time. The naïve

210 method would be to build at first the list then later sort its elements by  
 211 standard integer sorting; however, it would charge an undesirable logarithmic  
 212 overhead.

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

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

224 With the proviso that in the present implementation of DETECT the PSEs  
 225 detected at time  $t$  are those ending at time  $t - 1$ , we specify what is involved  
 226 in the first iteration of the inner loop of ALL-PSEs. DETECT performs Line  
 227 5, outputting  $F$  (which in the first iteration is  $= \widetilde{L}_{t-1} \cup \widehat{L}_{t-1}$ ) together with  
 228  $\tau_{max}$ , the latest entry time of any member of  $\widehat{L}_{t-1}$ . We need now (Line 6) to  
 229 remove from  $F$  all edges having entry time at  $\tau_{max}$  - which is nothing else  
 230 just update  $\widehat{L}_{t-1}$  by deleting the first sublist from it. Following that, the  
 231 same treatment is reapplied to the reduced set, and this is iterated until all  
 232 elements of  $\widehat{L}_{t-1}$  have been considered.

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

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

245 **4. Complexity**

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

251 Turning to DETECT, we charge the sweep of row 0, which is needed to  
 252 sort edges by entry time to build  $\widehat{L}_{t-1}$ , to the entries  $\mathbb{M}[0, t - \sigma]$ ,  $\mathbb{M}[0, t - \sigma -$   
 253  $1]$ ,  $\dots$ ,  $\mathbb{M}[0, \tau_{min}]$  of  $\mathbb{M}$ , that correspond to the segment of one of the longest  
 254 leaving edge in  $X_{t-1}$ . Each such segment is charged at most once in this  
 255 way, while the invariant condition guarantees that the total size of the lists  
 256 collected in the process is bounded by  $|E_t| < |V|^2/2$ , whence the total cost  
 257 throughout the iterations is  $\Theta(|V|^2 \times T/p)$ . As is easy to see, the remainder  
 258 of operations are charged to the output. In particular, every rescanning of  
 259 the list  $L_{t-1}$  mandated by Line 6 of ALL-PSEs results in one new PSE being  
 260 published.

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

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

279 With this implementation  $|\tilde{E}|$  is to be interpreted as the average cardi-  
 280 nality of sets of edges  $\{E_0, E_1, \dots, E_{T-1}\}$  over the time-steps of the process.  
 281 Note that the size of the input is  $\Theta(|V| + T \times |\tilde{E}|)$ . Adding up through all

282 periods yields time  $O(|V| + T^2/\sigma \times |\tilde{E}|)$  plus the size of the output, and the  
 283 total space used is  $\Theta(|V| + T \times |\tilde{E}|)$ .

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

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

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

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

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

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

296 *distinct PSEs.*

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

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

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

300 Publishing each PSE explicitly at an average cost of  $\Theta(|\tilde{E}|)$  time results  
 301 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.  
 302 However, an easy implicit description is possible that encapsulates all PSEs  
 303 ending at the same time-steps, with the result of eliminating the extra  $|\tilde{E}|$   
 304 factor in these bounds. For this, observe that for any fixed period  $p$  the

305 subgraphs  $F_i$  associated to the PSEs  $P_i = \langle F_i, t - s_i + 1, s_i \rangle$  detected at time  
 306  $t$  form a chain  $F_1 \subset F_2 \subset \dots \subset F_m \subseteq L_t \subseteq E_t$ . To identify the whole chain, it  
 307 is enough to describe the edges of  $L_t$  in order of decreasing entry time and  
 308 the differences  $F_i \setminus F_{i-1}$ , for  $i = 2, \dots, m$ , and to keep track of their entry  
 309 times. With notations we used in Section 3 we have to report one-by-one the  
 310 sublists of  $\widehat{L}_{t-1}$  where the very last sublist is accompanied by the updated  
 311  $L_{t-1}$ .

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

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

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

## 314 5. Concluding Remarks

315 We presented an online algorithm to find all periodic subgraphs embed-  
 316 dings (PSEs) of a given dynamic network  $\mathcal{G}$  over  $T$  time-steps that runs faster  
 317 than previously available methods. In particular, we provide an optimal so-  
 318 lution based on an implicit description of the output subgraphs.

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

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

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

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

335 The following observation reduces checking only to a subset of the possible  
 336 values of  $t'$  and  $p'$ .

337 **Observation 7.** *Suppose that PSE  $\langle F, S(t, p, s) \rangle$  is subsumed by some other*  
338 *PSE. Then there exists a prime number  $\lambda$  such that  $\lambda$  divides  $p$  and for each*  
339  *$e \in F$  we have that  $e \in \bigcap_{k=0}^{s \times \lambda} E_{t+p'k}$  for  $p' = p/\lambda$ .*

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

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

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

353 **Acknowledgements.** The authors are indebted to F. Cicalese for many  
354 helpful discussions. This research was carried out in part while the authors  
355 worked together at the Alfréd Rényi Institute in Budapest, with support  
356 from the Hungarian Bioinformatics MTKD-CT-2006-042794, Marie Curie  
357 Host Fellowships for Transfer of Knowledge. Additional support for Alberto  
358 Apostolico was provided by the United States-Israel Binational Science Founda-  
359 tion (BSF) Grant No. 2008217, and by the Research Program of Georgia  
360 Tech. Péter L. Erdős was in part supported by the Alexander von Humboldt  
361 Foundation and the Hungarian NSF, under contracts NK 78439 and K 68262.  
362 Ervin Gyóri was in part supported by the Hungarian NSF, under contract NK  
363 78439. Cinzia Pizzi was in part supported by Progetto Cariparo: 2008-2011  
364 Systems Biology Approaches to Infer Gene and Protein Time-series Expres-  
365 sion Data, and Progetto Ateneo: 2008-2010 Computational Assessment of  
366 Protein-Protein Interaction Networks and Validation as a Guide for Modern  
367 System Biology.

## 368 References

- 369 [1] A. Apostolico, M. Barbares, C. Pizzi, Speedup for a periodic subgraph  
370 miner, Information Processing Letters 111 (2011) 521–523.

- 371 [2] C. Carpineto, G. Romano, *Concept data analysis: theory and applica-*  
372 *tions*, John-Wiley & Sons Inc., Chichester, England, 2004.
- 373 [3] N. Eagle, A. (Sandy) Pentland, *Reality mining: sensing complex social*  
374 *systems*, *Personal Ubiquitous Comput.* 10 (2006) 255–268.
- 375 [4] M.R. Garey, D.S. Johnson, *Computers and Intractability: A Guide to*  
376 *the Theory of NP-Completeness*, W. H. Freeman & Co., New York, NY,  
377 USA, 1979.
- 378 [5] J. Han, H. Cheng, D. Xin, X. Yan, *Frequent Pattern Mining: Current*  
379 *Status and Future Directions*, *Data Mining and Knowledge Discovery* 14  
380 (2007).
- 381 [6] P. Juang, H. Oki, Y. Wang, M. Martonosi, L.S. Peh, D. Rubenstein,  
382 *Energy-efficient computing for wildlife tracking: Design tradeoffs and*  
383 *early experiences with zebranet*, in: *ASPLOS-X: Proceedings of the 10th*  
384 *International Conference on Architectural Support for Programming Lan-*  
385 *guages and Operating Systems*, ACM Press, New York, NY, USA, 2002,  
386 pp. 96–107.
- 387 [7] M. Lahiri, T. Berger-Wolf, *Periodic subgraph mining in dynamic*  
388 *networks*, *Knowledge and Information Systems* 24 (2010) 467–497.  
389 10.1007/s10115-009-0253-8.
- 390 [8] N. Pasquier, Y. Bastide, R. Taouil, L. Lakhal, *Efficient mining of associ-*  
391 *ation rules using closed itemset lattices*, *Inf. Syst.* 24 (1999) 25–46.