Fast Computation of Dense Temporal Subgraphs

Shuai Ma, Renjun Hu, Luoshu Wang, Xuelian Lin, Jinpeng Huai

SKLSDE Lab, Beihang University & Beijing Advanced Innovation Center for Big Data Brain Computing, China

{mashuai, hurenjun, wangluoshu, linxl, huaijp}@buaa.edu.cn

ICDE 2017

April 19-22
San Diego, CA

Introduction

In this paper, we are to study the problem of finding dense subgraphs (FDS) on large temporal graphs.

Temporal graph G(V,E,F). A temporal graph is an undirected weighted graph with nodes and edges unchanged and edge weights varying with timestamps regularly and constantly, *e.g.*, road traffic networks. Note that G(V,E,F) with T timestamps essentially denotes a sequence <$G_1(V,E,F^1)$, …, $G_T(V,E,F^T)$> of T snapshots of a standard graph.

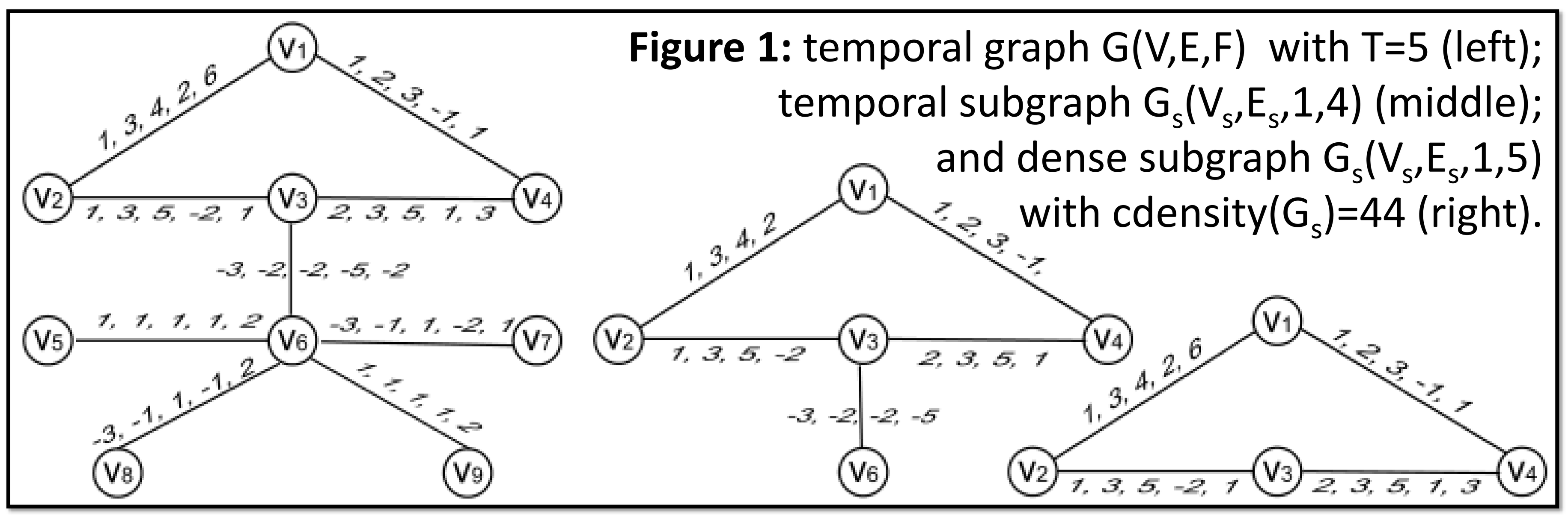


Figure 1: temporal graph G(V,E,F) with T=5 (left); temporal subgraph $G_s(V_s,E_s,1,4)$ (middle); and dense subgraph $G_s(V_s,E_s,1,5)$ with cdensity(G_s)=44 (right).

Temporal subgraph $G_s(V_s,E_s,i,j)$ of G(V,E,F). A temporal subgraph contains a subset of nodes and edges of G, and is restricted within the time interval that falls into [1,T]: $V_s \subseteq V$, $E_s \subseteq E$, $[i,j] \subseteq [1,T]$.

Finding dense subgraphs (FDS). FDS refers to *find a connected temporal subgraph with the greatest cohesive density*, where cohesive density cdensity(G_s)=$\sum F^t(e)$ for all $e \in E_s$ and $t \in [i,j]$, *i.e.*, the sum of edge weights among all snapshots.

Challenges and Limitations

Hardness: *The FDS problem is NP-hard to solve [1], and NP-hard to approximate within any constant factor, even for a temporal graph with a single snapshot and with +1 or -1 weights only.*

Limitations of filter-and-verification [1]. A temporal graph with T timestamps has a total of T*(T+1)/2 time intervals. The state-of-the-art solution [1] adopts a filter-and-verification framework that even if a large portion of time intervals (*e.g.*, 99%) are filtered, there often remain a large number of time intervals to verify.

T	141	447	1,414	…	14,142
T*(T+1)/2	10^4	10^5	10^6	…	10^8
# to verify	10^2	10^3	10^4	…	10^6

Filter-and-verification is insufficient for large temporal graphs!

A Data-driven Approach FIDES

We propose a highly efficient data-driven approach FIDES, instead of filter-and-verification, to finding dense temporal subgraphs.

Algorithm FIDES
Input: Temporal graph $\mathbb{G}(V, E, F)$, positive integer k.
Output: Subgraph of $\mathbb{G}$ with cohesive density as large as possible.
1. Identifying $k/2$ time intervals using maxTInterval;
2. Identifying $k/2$ time intervals using minTInterval;
3. **for** each $[i,j]$ of the k time intervals **do**
4. compute the dense subgraph of $\mathbb{G}[i,j]$ using computeADS;
5. **return** the subgraph with the largest cohesive density.

Procedure maxTInterval / minTInterval: identify k time intervals involved with dense subgraph by employing hidden data statistics.
Procedure computeADS: compute dense subgraph given time intervals.

Hidden Data Statistics and Time Intervals

Hidden data statistic ECP. The evolving convergence phenomenon (ECP) asserts that all edge weights evolve in a convergent way, *i.e.*, $\exists e$ $F^t(e) <$ (or, $>$) $F^{t-1}(e)$ implies $\forall e$ $F^t(e) \leq$ (or, $\geq$) $F^{t-1}(e)$. The ECP is inspired by the convergent evolution in evolutionary biology which describes that different species independently evolve similar traits as a result of having to adapt to similar environments (Fig. 2).

Figure 2: convergent evolution

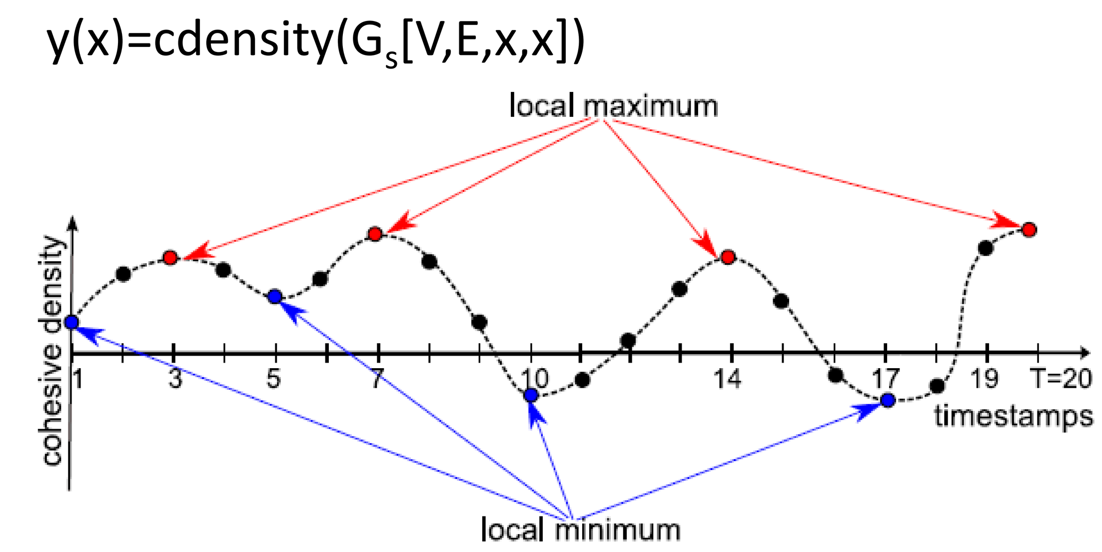


Figure 3: cohesive density curve

Characteristic under ECP: *to find the dense subgraphs, we only need to consider the time intervals [i,j] such that the cohesive density curve has a local maximum between i and j (Fig. 3).*

Procedure maxTInterval / minTInterval identify the top-k time intervals containing local maxima and having the largest positive cohesive density.

Procedure computeADS

Given [i,j], an aggregate graph G'(V,E,f) is first constructed *s.t.* $f(e)=\sum F^t(e)$ for $t \in [i,j]$ (Fig. 4(a)). Finding dense subgraph on G' can be reduced to the net worth maximization problem on a converted graph (Fig. 4(b)). And we further solve it with three optimization techniques: strong merging (Fig. 4(c)-4(d)), strong pruning (Fig. 4(e)) and bounded probing (Fig. 4(f)).

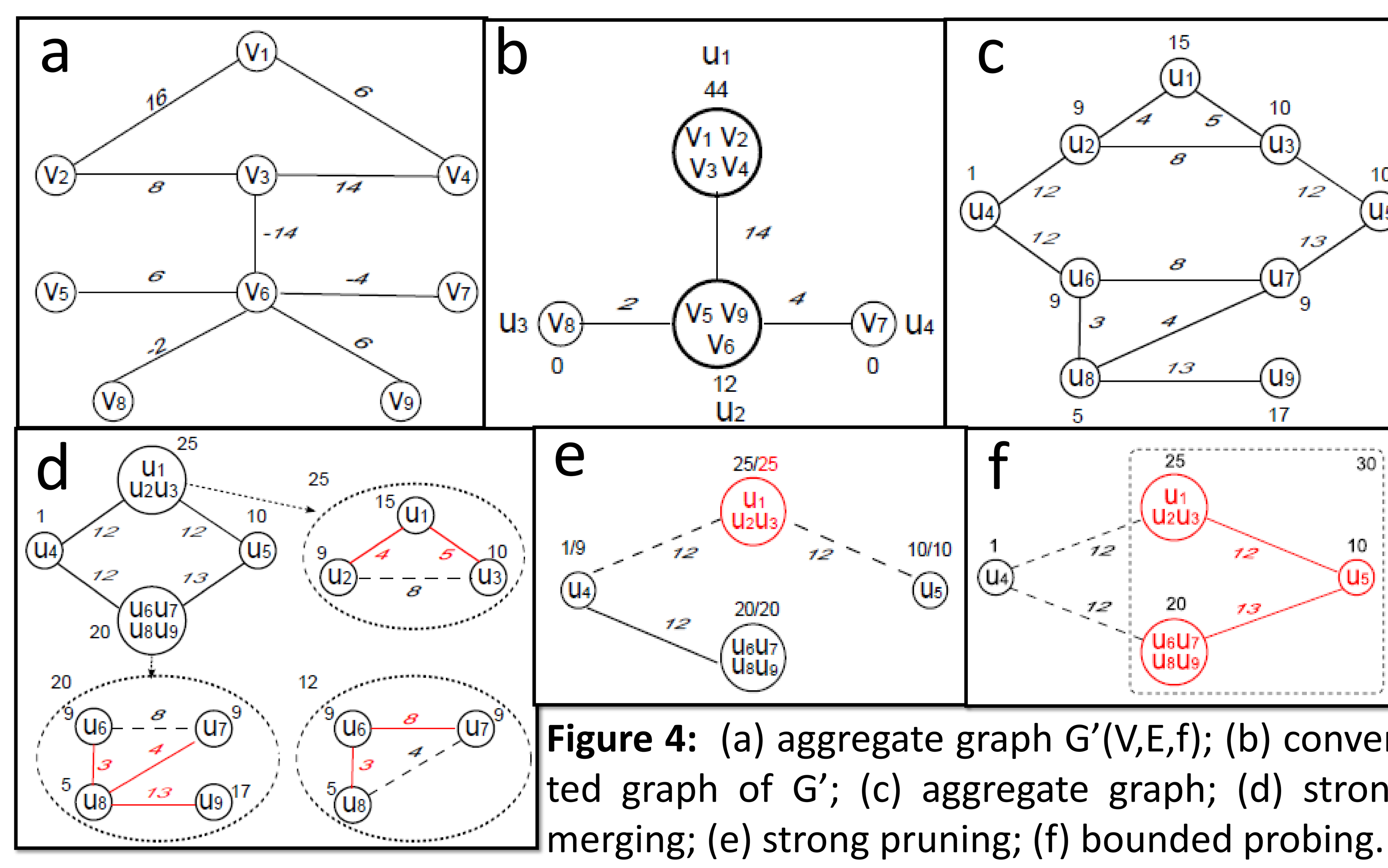
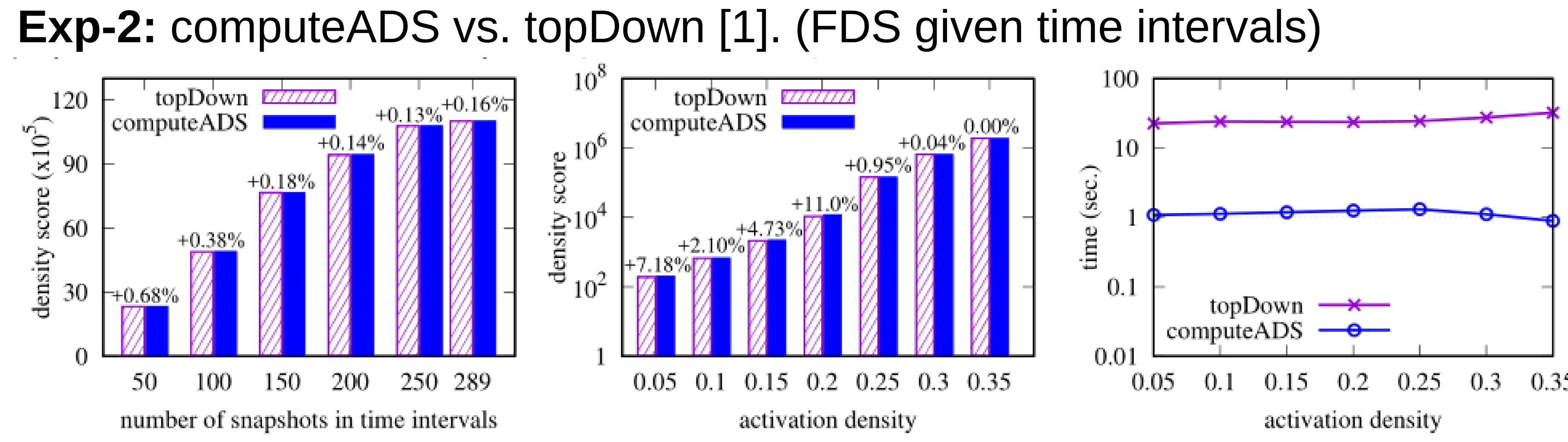


Figure 4: (a) aggregate graph G'(V,E,f); (b) converted graph of G'; (c) aggregate graph; (d) strong merging; (e) strong pruning; (f) bounded probing.

Experimental Results

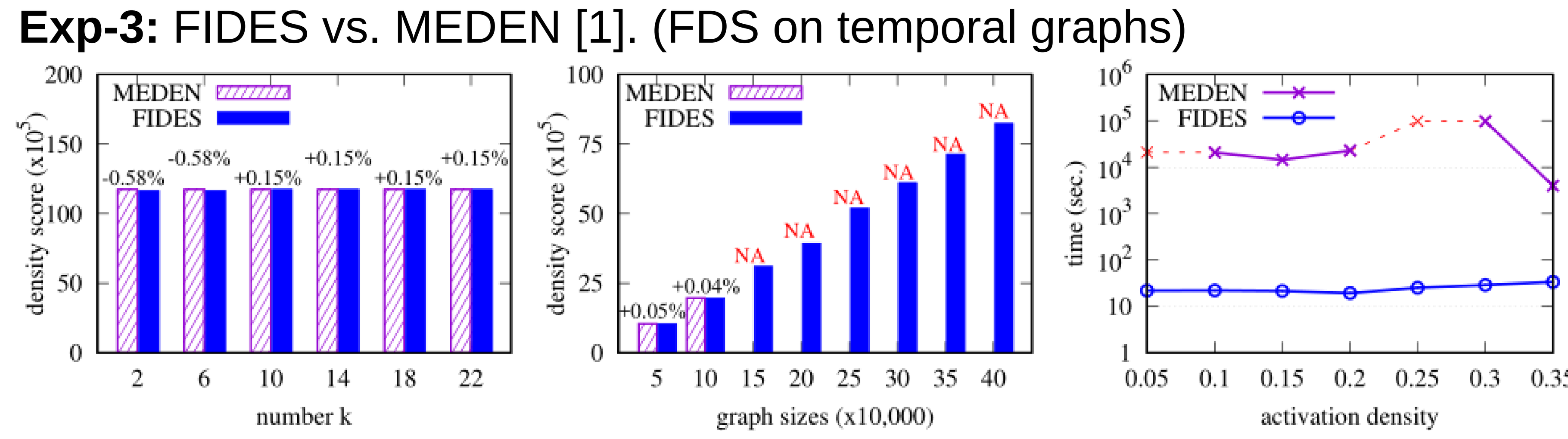
Exp-1: verification of ECP. The proportion of edges that satisfy ECP is 96% on BJData and 90% on average on SYNData.

Exp-2: computeADS vs. topDown [1]. (FDS given time intervals)



(a) BJData (b) SYNData (c) SYNData

Exp-3: FIDES vs. MEDEN [1]. (FDS on temporal graphs)



(a) BJData (b) SYNData (c) SYNData

Summary. *(1) ECP is quite common. (2) Algorithm computeADS performs better than topDown both in quality and efficiency. (3) FIDES finds comparable dense subgraphs to MEDEN, even with small k, and is 1000x faster than MEDEN.*

[1] P. Bogdanov, M. Mongiov, A. K. Singh. Mining heavy subgraphs in time-evolving networks. In *ICDM*, 2011.

This work is supported in part by NSFC U1636210, 973 Program 2014CB340300, NSFC 61421003&61322207, Special Funds of Beijing Municipal Science & Technology Commission, and MSRA Collaborative Research Program.