2017 IEEE 33rd International Conference on Data Engineering

BEAMS: Bounded Event Detection in Graph Streams

Mohammad Hossein Namaki*, Keyvan Sasani*, Yinghui Wu*, Tingjian Ge†
*Washington State University, †University of Massachusetts, Lowell
*{mnamaki, ksasani, yinghui}@eecs.wsu.edu, †ge@cs.uml.edu

***Abstract*—This demo presents BEAMS, a system that automatically discovers and monitors top-k complex events over graph streams. Unlike conventional event detection over streams of items, BEAMS is able to (1) characterize and detect complex events in dynamic networks as graph patterns; and (2) perform online event discovery with a class of *bounded* algorithms that compute changes to top-k events in response to the transactions in graph streams, and incurs a minimized time cost determined by the changes, independent of the size of graph streams. We demonstrate : a) how BEAMS identifies top-k complex events as graph patterns in graph streams, and supports ad-hoc event queries online, b) how it copes with the sheer size of real-world graph streams with bounded event detection algorithm; and c) how the GUI of BEAMS interacts with users to support ad-hoc event queries that detect, browse and inspect trending events.**

Video: https://youtu.be/lVUGM0Fa17Q

I. Introduction

Event detection over graph streams has found prevalent use in dynamic social networks, communication networks, and cyber security [3]. A graph stream $\mathcal{G}_T$ is a (possibly infinite) stream of graphs $\{G_0, \ldots, \}$ where each graph G_i is a "snapshot" of the evolving $\mathcal{G}_T$ at timestamp t. Given $\mathcal{G}_T$, a configuration of event models and quality measures (Fig. 3), the problem is to track top-k events in $\mathcal{G}_T$, and to provide answers to ad-hoc queries upon requests.

The need for online event detection motivates us to develop BEAMS, an event detection system over graph streams.

Example 1: The Offshore dataset [1] covers 40 years of offshore entities and their financial activities. BEAMS reports two top active events (Fig. 1), where P_1 identifies *"most active offshore jurisdiction (tax heavens) (variable x) of Asian companies since 1965"*, and specifies "British Virgin Island (BVI)" with 8294 active entities as the most active one; and P_2 finds a significant move of active bearer shares companies in 2005 from "BVI" to "Panama", due to that the former cracked down on bearer shares. A third active event P_3 for point of interests recommendation states that *"if a user x and his friends y who have checked in at a bar (via* e.g., *Facebook Place) retweet each other, and x checks in at a nearby sport club, then he is likely to visit the same bar (as his friends).* □

Complex events over networks are often characterized as graph patterns [3], [5] (as illustrated in Fig. 1). The detection and tracking of such events are more involved than their counterparts over item streams [2]. Temporal graph pattern matching [3] and mining [5] have been studied to find events

[1] https://offshoreleaks.icij.org/pages/database

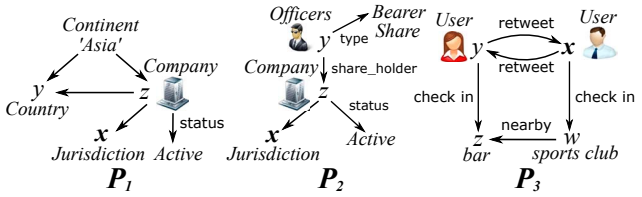

Fig. 1: Event patterns

as temporal graph patterns, and to identify active entities in dynamic networks, respectively. By contrast, BEAMS has the following unique features not addressed in prior work.

1) BEAMS supports online event detection as *general graph patterns*. To balance the expressiveness of event model and the cost of online mining, it uses (quadratic-time) approximate pattern matching to characterize complex events and their occurrences, rather than strict queries that require (NP-hard) subgraph isomorphism [5].

2) BEAMS supports *bounded* event detection in response to new transactions in a graph stream $\mathcal{G}_T$. Given a pair of consecutive snapshots G_i and G_{i+1}, BEAMS incrementally updates the top-k events by automatically tracking the affected patterns and their matches, characterized as *affected area* (AFF), which are necessarily checked by a batch counterpart. BEAMS incurs a cost bounded by a function of the size of AFF only, independent of the size of $\mathcal{G}_T$. That is, it performs only "necessary" amount of work to correctly update the events.

3) BEAMS provides user-friendly, easy-to-configure GUI to support ad-hoc event queries over graph streams.

II. Incremental Event Detection

We overview the event model and the incremental mining paradigms of BEAMS, and then introduce its architecture.

Event patterns. Unlike conventional systems, BEAMS characterizes an event as a graph pattern P that contains entities and their temporal relationships. It also supports user-specified "focus labels" (*e.g.,* "Asia" in Example 1) in P to discover events that pertain to the labels (*e.g.,* P_1 in Fig. 1). BEAMS supports approximate graph pattern matching to capture approximate matches of P. By default, it uses dual-simulation [3].

Top-k event mining. BEAMS supports practical need to discover k most "frequent" events. Given $\mathcal{G}_T$ over time interval $[1, T]$, the support of an event pattern P in $\mathcal{G}_T$ (denoted as $\mathsf{supp}(P, \mathcal{G}_T)$) is defined as $\sum_{i \in [1,T]} \alpha^{T-i} \frac{|\mathsf{occ}(P(\bar{u}),i)|}{|V_i|}$, where $\mathsf{occ}(P(\bar{u}), i)$ refers to the match set of the focus $\bar{u}$ of P in snapshot G_i, and $\alpha \in (0, 1]$ refers to a decay factor. Intuitively, $\mathsf{supp}(P, \mathcal{G}_T)$ quantifies the frequency of P by aggregating the

2375-026X/17 $31.00 © 2017 IEEE
DOI 10.1109/ICDE.2017.189

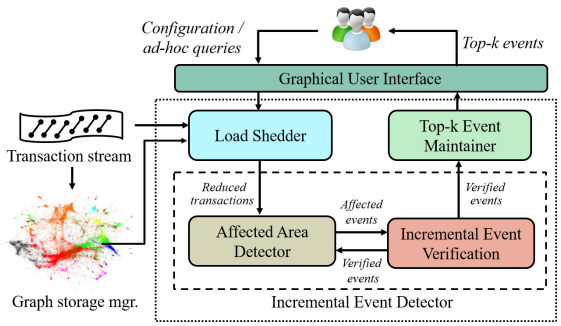

Fig. 2: Architecture of BEAMS

match size over each snapshot G_i, with "discounted" effect of past data. We are interested in detecting "maximal" event patterns, for which no larger patterns are frequent.

Given graph stream $\mathcal{G}_T$, and a configuration of (optional) parameters including: support threshold θ, focus $\bar{u}$, decay factor α, integer k, and an event size bound b_p, BEAMS tracks top-k maximal frequent events Σ pertaining to $\bar{u}$, θ and b_p such that the total support $\sum_{P_i \in \Sigma} \mathsf{supp}(P_i)$ is maximized.

Bounded algorithm. At the core of BEAMS is an incremental event detector, which incrementally updates top-k events by processing edge updates in batches. (1) The detector "cold starts" the online detection by invoking a batch pattern mining algorithm to compute top-k events Σ_0 over the first snapshot G_0 from scratch. (2) For each batch updates Δ_i that changes G_i to G_{i+1}, it invokes a *bounded* incremental mining algorithm to update Σ_i to Σ_{i+1}. To this end, it dynamically determines a set of pattern candidates that need to be re-verified due to edge updates in G_i (*e.g.,* having candidates within d-hop of the updates in G_i), as well as their candidate matches in G_i. This fraction of data is characterized as *affected area* (AFF), which are necessarily checked by any batch algorithm in order to update Σ_i at any timestamp i.

By performing incremental verification for affected patterns, the overall cost is determined by a function of the size of AFF, independent with the size of G_i and $\mathcal{G}_T$.

III. System Architecture

The architecture of BEAMS (shown in Fig. 2) contains four modules. (1) *Graphical User Interface* (GUI) takes mining configurations from users, and returns visualizations to demonstrate the top-k events (Fig. 3). (2) A *Load shedder* applies the principle of load shedding [4] to reduce transactions at runtime for affected events, (3) The *Incremental event detector* dynamically maintains affected area AFF with *Affected area detector*, and incrementally verify the affected patterns (*Incremental Event Verifier*). (4) The verified patterns are fed to *top-k event maintainer*, which are used to update top-k event set and visually interpreted at GUI.

We implemented BEAMS in Java with the property graph model[2]. The source code is available at https://goo.gl/g0erkj. An online demo is available at https://goo.gl/nsuAkV.

IV. Demonstration Overview

Datasets. We use (1) IDS[3], which records daily intrusion activity over a cyber network with 33M entities (*e.g.,* alert, host, logs) and 144M relations (*e.g.,* application, protocol); (2) Citation, a citation network[4] with 4.3M entities and 21.7M edges (*e.g.,* citation, published at); and (3) Offshore (Example 1) with 839K business entities and 3.6M financial relations.

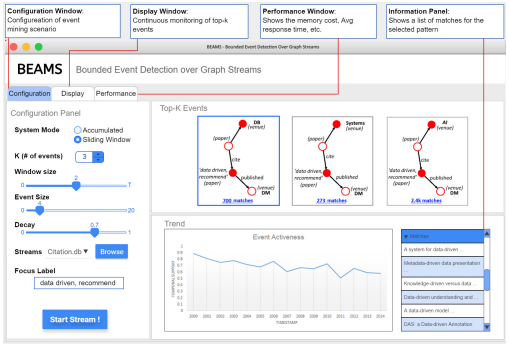

Fig. 3: GUI: Event Monitoring and Ad-hoc Queries

Ease of use. We invite the users to play with the user-friendly GUI of BEAMS (see Fig. 3) to experience an interactive event detection scenarios. Users are also able to track the trend of specific events over time via the display panel.

Performance of bounded event detection. We demonstrate the performance of BEAMS compared with (1) its batch counterpart, which rediscovers the events from scratch; and (2) a variant that uses GraMi to mine frequent subgraphs as events [1]. We show that BEAMS is feasible over large networks: it takes on average 6.7 seconds upon single batch of updates of $40K$ transactions, and outperforms its batch counterpart by 52.93 times. On the other hand, its GraMi-based counterpart does not terminate over graphs of 10M entities after 10 hours.

Ad-hoc event queries. We invite users to retrieve interesting events with ad-hoc queries. Two example queries are: (1) *Most active tax heavens of Asian companies*, where the top-3 events discovered by BEAMS over Offshore specifies "*British Virgin Island*, "*Panama*" and "*Bahamas*" as active tax heavens; and (2) *Top venues with most papers that cited targeted data mining topics in the last 5 years*. BEAMS reports a "trace" of an evolving citation pattern with focus "data-driven recommendation" over Citation from 2000 to 2015 (Fig. 3).

Acknowledgments. Namaki and Wu are supported in part by NSF IIS-1633629 and Google Faculty Research Award. Tingjian Ge is supported in part by the NSF, under the grants IIS-1149417, IIS-1319600, and IIS-1633271.

References

[1] Y. W. Qi Song and X. L. Dong. Mining summaries for knowledge graph search. In *ICDM*, 2016.

[2] J. F. Roddick and M. Spiliopoulou. A survey of temporal knowledge discovery paradigms and methods. *TKDE*, pages 750–767, 2002.

[3] C. Song, T. Ge, C. Chen, and J. Wang. Event pattern matching over graph streams. *PVLDB*, 8(4):413–424, 2014.

[4] N. Tatbul, U. Çetintemel, S. Zdonik, M. Cherniack, and M. Stonebraker. Load shedding in a data stream manager. In *VLDB*, 2003.

[5] B. Zong, X. Xiao, Z. Li, Z. Wu, Z. Qian, X. Yan, A. K. Singh, and G. Jiang. Behavior query discovery in system-generated temporal graphs. *VLDB*, 9(4):240–251, 2015.

[2] https://neo4j.com

[3] http://www.unb.ca/research/iscx/dataset/iscx-IDS-dataset.html

[4] https://aminer.org/citation