A study on Historical Web Graph Extraction

Zhenglu Yang and Masaru Kitsuregawa

Institute of Industrial Science

University of Tokyo

Abstract

Discovery of evolving regions in large graphs is an important issue because it is the basis of many applications such as spam websites detection in the Web, community lifecycle exploration in social networks, and so forth. In this paper, we aim to study a new problem, which explores the evolution process between two historic snapshots of an evolving graph. The evolution process is simulated as a fire propagation scenario based on the Forest Fire Model. We propose two efficient solutions to tackle the issue which are grounded on the probabilistic guarantee. The experimental results show that our solutions are efficient with regard to the performance and effective on the well fitness of the major characteristics of evolving graphs.

1. Introduction

Graphs represent the complex structural relationships among objects in various domains in the real world. While these structural relationships are not static, graphs evolve as time goes by. Evolving graphs are usually in the form of a set of graphs at discontinuous time stamps, where the period between two adjacent time stamps may be quite long. Take the Web archive for example. Due to its large size, the Web or a part of it is periodically archived by months or even by years.

While many of the existing studies have paid attentions to finding stable or changing regions in evolving graphs [1,3,5], only a few of them are about how graphs evolve. In this paper, we study a new problem, which is to model the evolving process between two historical snapshots of an evolving graph. Fig. 1 briefly shows our idea. G is an evolving graph which evolves from time *t* to *t'*. Suppose we have the graph snapshots at time t and *t'*, and the real evolution details between these two snapshots are unknown. We would like to generate a series of virtual graph snapshots, as shown in the dotted parts in Fig. 1.

Our approach adopts the Forest Fire Model

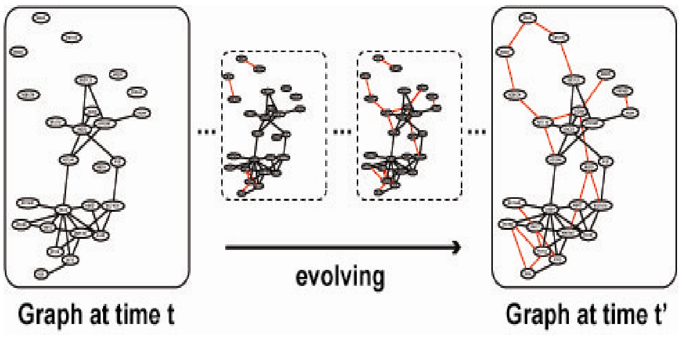

Fig. 1 Example Graphs

(FFM) [4], which has been demonstrated as successfully explaining the mechanism of dynamic systems [2]. The new edge linkage action can be thought of as fire propagation, and the nodes on the new edges are the origins of the fire. The virtual historical graph is then to be thought of as the snapshot on tracking how the fire propagates on the whole graph.

2. Discovery of the Historical Snapshot Graph Problem

Given two graphs, G_t and G_{t+1}, at time *t* and time *t+1*, the problem of discovering the historical graph snapshots involves tracing back the virtual graphs after inserting *n* new edges, where $1 \leq n \leq (|E(G_{t+1})| - |E(G_t)|) = k$, into the old graph G_t. In this paper, the issue of finding *n*-graphs is equivalent to discovering the burning out n-graphs, which is defined as follows:

Definition 1: Burning Out n-Graph (BOG)
Given two undirected graphs $G_t=(V_t, E_t)$ and $G_{t+1}=(V_{t+1}, E_{t+1})$; a cost function $CF(i, j)$ on edge (i, j) where $i \in V_t \cup V_{t+1}$ and $j \in V_t \cup V_{t+1}$; a user preferred number *n* of new edges, find the subgraph *H* of $G_t \cup G_{t+1}$ such that

- The number of new edges on *H* is *n*
- $\sum_{(i,j) \in E(H)} CF(i, j)$ is minimized.

3. Proposed Approaches

In this paper, we propose two approaches to discover the fastest burning out *n*-graphs. The bottom-up approach examines the candidates from scratch in a global view with the dynamic threshold guaranteed, while the leap-search approach proposes a density-oriented candidate selection strategy.

3.1 The Bottom-Up Approach

We develop a bottom-up greedy algorithm to extract the burning out n-graphs. The algorithm follows the candidate generation and verification iteration. To accelerate the process, the threshold-based technique is proposed to prune the candidates early.

Candidate generation: In each iteration, a *k*-graph *g* is grown up to a candidate (*k*+1)-graph *g'* by introducing a new edge e_{new}, where the following conditions hold: (1) e_{new} is connected to some vertices in *g*; and (2) the fire spreading time *t* from *g* to v_{new} is minimized.

Verification: If the burning out time of the candidate (*k*+1)-graph *g'* is greater by far than the best one, we turn to the next candidate subgraph. Otherwise, we update the fire energy of the new vertex and continually grow *g'* to a larger candidate graph with another candidate-generation-and-test iteration.

3.2 The Leap Search Approach

We present a leap-search based method for the extraction of burning out *n*-graphs. The method is efficient in processing graphs with a large *n*, where growing up the candidate subgraphs from scratch by using bottom-up growth can be time consuming. The approach is based on the density-oriented candidate selection strategy. The intuitive idea is that fire transfers fast in those regions where the energy density is high. The extraction process is also composed of candidate generation and verification iteration.

Candidate generation: Starting from the nodes with the most fire energy, we grow them by selecting their nearest neighbors (w.r.t. the fire propagation time) until the number of the new edges in the subgraph is equal to *n*. In other words, we do not wait for other possible candidates to grow up. During the growing process, we record the least time necessary to transfer the fire.

Verification: We check the bottleneck nodes of the fire propagation in these subgraphs (as indicated by the least time), greedily find the neighbors which can remedy the weak edges on spreading the fire, and then update the least time value. Through a recursive process, we finally determine the first burning out regions with the least *n* new edges.

We propose several efficient strategies to fasten the process for the two approaches. Both algorithms only need to test and update a small number of candidate graphs, which lead to good scalability with respect to the cardinality of the datasets and the value of *n*.

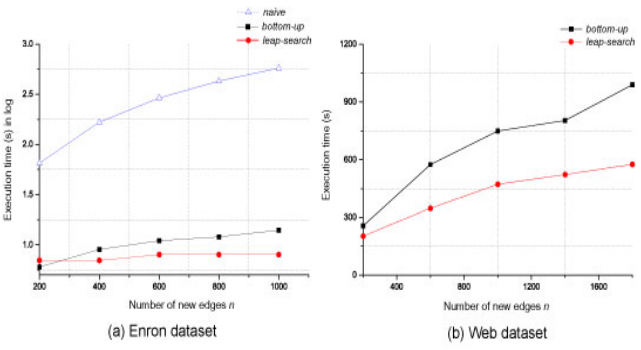

Fig. 2 Evaluation on efficiency

4. Experiments

We compare our two algorithms, bottom-up and leap-search, with a naive method. The result on efficiency is shown in Fig. 2. Note that the execution time on the *Enron* dataset is in logarithm format. We can see that our solutions are much faster than the naive one, about one to two orders of magnitude (as shown in Fig. 2 (a)). For the huge Web dataset, the naive algorithm can not finish in reasonable time. Between our two approaches, the leap-search performs better than the bottom-up when the number of new edges *n* becomes larger. The reason is that for large value of *n*, growing graphs from scratch by evaluating all the neighbors is time consuming.

Reference

[1] N. Bansal, F. Chiang, N. Koudas, and F. W. Tompa. Seeking stable clusters in the blogosphere. In *VLDB*, 2007.
[2] C. Henley. Self-organized percolation: A simpler model. *Bull. Am. Phys. Soc.*, 1989.
[3] A. Inokuchi and T. Washio. A fast method to mine frequent subsequences from graph sequence data. In *ICDM*, 2008.
[4] R. C. Rothermel. A mathematical model for predicting fire spread in wildland fuels. USDA Forest Service, Ogden, UT, Tech. Rep., 1972.
[5] J. Sun, C. Faloutsos, S. Papadimitriou, and P. S. Yu. Graphscope: parameter-free mining of large time-evolving graphs. In *KDD*, 2007.