Real-Time Diameter Monitoring for Time-Evolving Graphs

Yasuhiro Fujiwara^1, Makoto Onizuka^1, and Masaru Kitsuregawa^2

^{1} NTT Cyber Space Laboratories
^{2} Institute of Industrial Science, The University of Tokyo

Abstract. The goal of this work is to identify the diameter, the maximum distance between any two nodes, of graphs that evolve over time. This problem is useful for many applications such as improving the quality of P2P networks. Our solution, G-Scale, can track the diameter of time-evolving graphs in the most efficient and correct manner. G-Scale is based on two ideas: (1) It estimates the maximal distances at any time to filter unlikely nodes that cannot be associated with the diameter, and (2) It maintains answer node pairs by exploiting the distances from a newly added node to other nodes. Our theoretical analyses show that G-Scale guarantees exactness in identifying the diameter. We perform several experiments on real and large datasets. The results show that G-Scale can detect the diameter significantly faster than existing approaches.

Keywords: Diameter, Graph mining, Time-evolving.

1 Introduction

Graphs arise naturally in a wide range of disciplines and application domains. The distances between pairs of nodes are a fundamental property in graph theory. The node-to-node distances are often studied in terms of the **diameter**, the maximum distance in a graph. However, the focus of traditional graph theory has been limited to just *static* graphs; the implicit assumption is that the number of nodes and edges never change.

Recent years have witnessed a dramatic increase in the availability of graph datasets that comprise many thousands and sometimes even millions of time-evolving nodes; this is one consequence of the widespread availability of electronic databases and the Internet. Recent studies on large-scale graphs are discovering several important principles of time-evolving graphs [12]. Thus demands for efficient approaches to the analysis of time-evolving graphs are increasing.

In this paper, we focus on the problems faced when attempting to identify the exact diameter of a graph evolving over time by the addition of nodes. In other words, the goal of this work is continuous diameter monitoring for time-evolving graphs. We propose an algorithm to solve this problem exactly in real-time. The commonly-used approach to diameter computation is based on breadth-first search, which is not practical for large-scale graphs since it requires excessive CPU time. To the best of our knowledge, this is the first study to address the diameter detection problem that guarantees exactness and achieves efficiency.

1.1 Contributions

We propose a novel method called G-Scale that can efficiently identify the diameter of time-evolving graphs. In order to reduce monitoring cost, (1) we estimate the maximal distance to prune unlikely nodes that cannot be associated with the diameter, and (2) we maintain the answer node pairs whose distances are the diameter by exploiting distances from a newly added node to other nodes. G-Scale has the following attractive characteristics:

– **Efficient:** G-Scale is drastically faster than the existing algorithm. The existing algorithm takes $O(n^2 + nm)$ time where n and m are the number of nodes and edge, respectively, and so is prohibitively expensive for large-scale graphs.
– **Exact:** G-Scale does not sacrifice accuracy even though it prunes unlikely nodes in the monitoring process; it can exactly track the node pair that delimit the diameter of a time-evolving graph at any time.
– **Parameter-free:** Previous approximate approaches require the setting of parameters. G-Scale, however, is completely automatic; this means it does not require the user to set any parameters.

1.2 Problem Motivation

The problem tackled in this paper must be overcome to develop the following important applications. The network architecture called P2P is the basis of several distributed computing systems such as Gnutella, Seti@home, and OceanStore [2]. And content distribution is a popular P2P application on the Internet. For example, Kazaa and its variants have grown rapidly over time, over 4.5 million users share a total of 7 petabytes of data [3]. In a content distribution network, personal computers can use hop-by-hop data forwarding between themselves. An important and fundamental question is how many neighbors should a computer have, i.e., what size the routing table should be [16]. This question is important for two reasons. The number of computers in a P2P network could be extremely large, hence the complete routing table is likely to be too large to maintain. Second, because each hop in a P2P network is overhead, suppressing the query hop number by increasing table size is important in raising service efficiency.

Network diameter is a useful metric when trying to raise the search efficiency of a content distribution network, since it directly corresponds to the number of hops a query needs to travel in the worst case [9]. Moreover, by monitoring network diameter, the routing table size can be updated more effectively; if the diameter is large, the routing table size should be increased. This strategy can bound the search speed of content distribution networks.

In addition to the application presented above, robustness improvement is an important application in P2P networks of diameter monitoring. Koppula et al. showed that it can be determined which edges should be added/rewired to improve network robustness by plotting the diameters of dynamically changing graphs [8]. Furthermore, our proposed method can be used in other applications such as measuring the structural robustness of metro networks [13], monitoring the evolution of the Internet [11], and measuring citation networks size [10]. While time-evolving graphs are potentially useful in many applications, they have been difficult to utilize due to their high computational costs. However, by providing exact solutions in a highly efficient manner, G-Scale will allow many more data mining applications based on time-evolving graphs to be developed in the future.

The remainder of this paper is organized as follows. Section 2 describes related work. Section 3 overviews some of the background of this work. Section 4 introduces the main ideas of G-Scale and explains its algorithm. Section 5 gives theoretical analyses of G-Scale. Section 6 reviews the results of our experiments. Section 7 is our brief conclusion.

2 Related Work

Many papers have been published on approximation for node-to-node distances. The previous distance approximation schemes are distinguished into two types: annotation approach and embedding approach. Rattigna et al. studied two annotation schemes [17]. They randomly select nodes in a graph and divide the graph into regions that are connected, mutually exclusive, and collectively exhaustive. They give a set of annotations to every node from the regions. Distances are computed by the annotations. They demonstrated their method can compute node distances more accurately than the embedding approaches.

The Landmark approach is an embedding approach [7,15], and estimates node-to-node distance from selected nodes. The minimum distance via a landmark node is utilized as node distance in this method. Another embedding approach is Global Network Positioning which was studied by Ng et al [14]. Node distances are estimated with L_p norm between node pairs.

However, interest of these approaches lies only in the estimation; these approaches do not guarantee exactness.

3 Preliminary

In this section, we formally define some notations and introduce the background to this paper. Content distribution networks and others can be described as graph $G = (V, E)$, where V is the set of nodes, and E is the set of edges. We use n and m to denote the number of nodes and edges, respectively. That is $n = |V|$ and $m = |E|$. We define a **path** from node u to v as the sequence of nodes linked by edges, beginning with node u and ending at node v. A path from node u to v is the **shortest path** if and only if the number of nodes in the path is the smallest possible among all paths from node u to v. We use $d(u, v)$ to denote the **distance** between node u and v, and $d(u, v)$ is the number of edges in the shortest path from node u to v in the graph. By definition, $d(u, u) = 0$ for every $u \in V$, and $d(u, v) = d(v, u)$ for $u, v \in V$.

The diameter,D, is defined as the maximal distance between two arbitrary nodes as follows [5]: $D = \max(d(u, v)|u, v \in V)$. And our algorithm returns not only the diameter but the node pairs whose distances are equal to the diameter, $\mathcal{D}$. $\mathcal{D}$ is formally defined as follows: $\mathcal{D} = \{(u, v)|d(u, v) = D\}$.

The diameter of graph G can exactly be computed by the breadth-first search approach [4]. But the breadth-first search based approach generally needs $O(n^2 + nm)$ time because it computes the distances from all n nodes in a graph and $O(n + m)$ time is required for each node [6]. This incurs excessive CPU time for large-scale graphs as illustrated by the statement 'computing shortest paths among all node pairs is computationally prohibitive' made in [10]. Furthermore, the naive approach to monitoring time-evolving graphs is to perform this procedure each time a graph changes. However, considering the high frequency with which graphs evolve, a much more efficient algorithm is needed.

4 Monitoring the Diameter

In this section, we introduce the two main ideas and describe the algorithm of G-Scale. The main advantage of G-Scale is that it can efficiently and exactly solve the problem of identifying the diameter of time-evolving graphs. First we give an overview of each idea and then a full description.

4.1 Ideas Behind G-Scale

Our solution is based on the two ideas described below.

Reference node filtering. Our first idea is to prune unlikely nodes efficiently so as to reduce the high cost of the existing approach. The existing algorithm requires high computation time because it computes distances for all pairs of n nodes in the graph. Our idea is simple; instead of computing distances from all nodes, we compute the distances only from selected nodes and prune unlikely nodes. In other words, we use selected nodes to filter unlikely nodes. We refer to the selected nodes as **reference nodes**.

In the monitoring process, we select reference nodes one by one and compute the distances from the node to other nodes. In doing so, we estimate whether nearby nodes of the reference node can delineate the diameter. The time incurred to estimate node distances is $O(1)$ for each node. As a result, if the number of reference nodes is k ($k \ll n$), $O(kn+km)$ time is required to detect the diameter, instead of the $O(n^2 + nm)$ time required by the existing algorithm solution.

This new idea has the following two major advantages. First, we can identify the diameter exactly even though we prune nearby nodes with the estimation. This means that we can safely discard unlikely nodes at low CPU cost. Note that the number of k is automatically determined. Generally, it is difficult to set parameters which would significantly impact the final result. Our approach, however, avoids user-defined parameters, and this is the second advantage.

Incremental update. Time-evolving graphs evolve by the addition of nodes over time. By a node addition, the diameter can *grow*, *shrink*, or be *unchanged*. We propose an algorithm that efficiently maintains the answer node pairs to detect the diameter of time-evolving graphs.

The naive method based on the above filtering approach for time-evolving graphs is to identify the diameter by setting reference nodes every time a node added. However, we ask the question, 'Can we avoid re-estimating the maximal distance every time the graph grows?'. This question can be answered by examining whether the node addition changes the answer node pairs. As described in detail later, if the diameter does not shrink with node addition, the answer node pairs can be incrementally updated by assessing *only* the distances from the added node.

This idea is especially effective for time-evolving graphs. In the case of time-evolving graphs, a small number of new nodes are continually being added to the large number of existing nodes. Therefore, there is little difference in the graphs before and after the addition of nodes, even if the new nodes arrive frequently. As a result, we can efficiently update the diameter and the answer node pairs by computing distances from the added node.

4.2 Reference Node Filtering

Our first idea involves selecting reference nodes so as to filter unlikely nodes efficiently.

Our filtering algorithm is as follows: (1) It computes the candidate distance which is expected to be diameter. (2) It selects reference nodes and estimates the maximal distances of nearby nodes to other nodes, and (3) If a node distance estimation yields a shorter distance than the candidate distance, it prunes the node since the node cannot be delineate the diameter. Accordingly, the unlikely node can be filtered quickly.

In this section, we first describe how estimate the maximal distance of a node to other nodes, and show that node distance estimation gives an upper bound for the maximal distance. We then introduce our approach of selecting the reference nodes and computing the candidate distance.

Formally, the following equation gives the maximal distance of node u: $d_{max}(u) = \max(d(u,v)|v \in V)$. We then define the estimation of the maximal distance as follows:

Definition 1 (Estimation). *For graph G, let u_r be a reference node, we define the following estimation of the maximal distance of node v, $\hat{d}_{max}(v)$, to filter unlikely nodes:*

$$\hat{d}_{max}(v) = d_{max}(u_r) + d(u_r, v) \tag{1}$$

We show the following lemma to introduce the upper bounding property of node estimation; this property enables G-Scale to identify the diameter exactly.

Lemma 1 (Upper bound). *For any node in graph G, the following inequality holds.*

$$d_{max}(v) \leq \hat{d}_{max}(v) \tag{2}$$

Algorithm 1. Filtering

```
Input: $G_t = (V, E)$, a time-evolving graph at time $t$
       $\mathcal{D}_{t-1}$, the answer node pairs of previous time tick
Output: $D_t$, the diameter of graph $G_t$
        $\mathcal{D}_t$, the answer node pairs
 1: $D_t := \max(d(u,v)|(u,v) \in \mathcal{D}_{t-1})$;
 2: $\mathcal{D}_t := \emptyset$;
 3: $V' := V$;
 4: while $V' \neq \emptyset$ do
 5:    $u_r := \text{argmax}(deg(v)|v \in V')$;
 6:    compute the maximal distance $d_{max}(u_r)$;
 7:    if $d_{max}(u_r) = D_t$ then
 8:       append $\{(u_r, v)|d(u_r, v) = D_t\}$ to $\mathcal{D}_t$;
 9:    end if
10:    if $d_{max}(u_r) > D_t$ then
11:       $D_t := d_{max}(u_r)$;
12:       $\mathcal{D}_t := \{(u_r, v)|d(u_r, v) = D_t\}$;
13:    end if
14:    subtract $u_r$ from $V'$;
15:    subtract $\{v|\hat{d}_{max}(v) < D_t\}$ from $V'$;
16: end while
```

Proof. Omitted for space. □

If a node estimation yields a shorter distance than the candidate distance, the node cannot be a node of the answer node pairs. So we prune the node. Since node estimation can be computed at the cost of $O(1)$ as shown in Definition 1, we can efficiently identify the diameter by exploiting node estimation.

Selection of the reference nodes and candidate distance are very important for efficient filtering; if the maximal distance of the reference node is longer than the maximal distance of the candidate node, we cannot effectively prune unlikely nodes (see Definition 1).

We select the highest-degree nodes as the reference nodes since the maximal distances of such nodes are likely to be short than other nodes; from such nodes, all nodes can be reached in a small number hops. We utilize the answer node pairs of the previous time tick to compute the candidate distance. Since graphs are almost the same after a node addition, the prior answer node pairs are expected to remain valid. These two techniques allow us to obtain good reference nodes and candidate distances effectively as demonstrated in the experiments in Section 6.

Algorithm 1 shows the filtering algorithm that detect the diameter by the reference nodes. The number of reference nodes, k, is automatically obtained in this algorithm. In this algorithm, $deg(u)$ represents the degree of node u. The algorithm first computes the candidate distance based on the prior answer node pairs (line 1). It then selects a reference node according to degree (line 5). If the maximal distance of the reference node is equal to the candidate distance, it appends the answer node pairs (lines 7-9). If the maximal distance of the reference node is larger than the candidate distance, it sets the candidate distance and the answer node pairs (lines 10-13). It uses the candidate distance to prune the unlikely nodes in the graph. That is, if a node distance estimation is less than the candidate distance, that node cannot delineate the diameter, and so can be safely discarded (line 15). This procedure is iterated until all nodes have been processed. This implies that the number of reference nodes, k, is automatically computed. That is, this algorithm does not require any user-defined parameters.

4.3 Incremental Update

Our second idea is an incremental monitoring algorithm that efficiently maintains the answer in case a node addition; it suppresses the computation time by providing conditions that restrict the application of the filtering algorithm for node addition.

Diameter changes. In this section, we first describe the property of node distance after node addition, and then examine the conditions in which the diameter grows, shrinks, or is unchanged. We assume that one node, u_a, and its connected edges are added to a time-evolving graph at each time tick.

We introduce below the property that underlies our update algorithm; distances of already existing node pairs can not be increased by node addition:

Lemma 2 (Distances after node addition). *Node distances at time t can not be longer than that at time $t-1$ for all node pairs in graph G_{t-1}.*

Proof. If all shortest paths between node u and v at time t pass through the added node, the corresponding path at time $t-1$ cannot have existed. That is, all the shortest paths at $t-1$ must be shortened by the added node. Otherwise, there exists a shortest path between node u and v at time t that does not pass through the added node. Therefore, the same path was present at time $t-1$. As a result, distance between node u and v is not increased by node addition. □

After node addition, the diameter can change. We distinguish three types of changes in diameter after node addition, and we theoretically analyze the three lemmas of the changes by utilizing the above property.

The first type of change is diameter increase. The diameter increases iff the maximal distance of the added node is longer than the diameter of the previous time:

Lemma 3 (Growth in diameter). *The diameter grows at time t if and only if:*

$$d_{max}(u_a) > D_{t-1} \tag{3}$$

Proof. If $D_t > D_{t-1}$, then node u_a must delineate the diameter since the maximal distances of already existing nodes cannot be longer than D_{t-1} from Lemma 2. If $d_{max}(u_a) > D_{t-1}$, then obviously $d_{max}(u_a) = D_t$ and $D_t > D_{t-1}$. □

The diameter shrinks iff the maximal distance of added node is shorter than the diameter at the previous time tick, and node addition invalidates all previous answer pairs:

Lemma 4 (Shrinkage in diameter). *The diameter shrinks at time t if and only if:*

$$\begin{aligned}&(1) d_{max}(u_a) < D_{t-1}, \textit{and}\\&(2) \forall(v,w) \in \mathcal{D}_{t-1}, d(v,w) < D_{t-1}\end{aligned} \tag{4}$$

Proof. If $D_t < D_{t-1}$, then $d_{max}(u_a) < D_{t-1}$ and all distances of answer nodes pairs at the last time tick must be shorter than D_{t-1}. If (1) and (2) hold, then the diameter shrinks at time t because of Lemma 2. □

The diameter would be unchanged after node addition iff the maximal distance of the added node is equal to the diameter of the previous time, or there exists at least one node pair whose distance is equal to the diameter at the previous time tick:

Lemma 5 (Unchanged diameter). *The diameter is unchanged at time t if and only if:*

$$\begin{aligned}&(1) d_{max}(u_a) = D_{t-1}, \textit{or}\\&(2) \exists(v,w) \in \mathcal{D}_{t-1} \;\; s.t. \;\; d(v,w) = D_{t-1}\end{aligned} \tag{5}$$

Proof. This is obvious from Lemma 3 and 4. □

Monitoring algorithm. We can efficiently maintain the answer with the incremental update approach. As the first step, we describe invalidation of answer pairs can be checked with only distances from added node, and then show our monitoring algorithm based on the incremental update approach.

We exploit the following property of the shortest path, which is shown in [4], to update the diameter and the answer node pairs:

Lemma 6 (Bellman criterion [4]). *Node u lies on a shortest path between node v and w, if and only if:*

$$d(u,v) + d(u,w) = d(v,w) \tag{6}$$

With Lemma 6, we can check whether node addition shortens distances of previous answer pairs:

Lemma 7 (Distance check). *In time-evolving graphs, the added node u_a shortens the distances of previous answer node pair (v,w) if and only if:*

$$d(v,u_a) + d(w,u_a) < D_{t-1} \tag{7}$$

Proof. If the added node u_a shortens the distances, then node u_a must lie on the shortest path between node v and w. Therefore, $D_{t-1} > d(v,w) = d(u_a,v) + d(u_a,w)$ from Lemma 6. If $d(v,u_a) + d(w,u_a) < D_{t-1}$, then the added node u_a must shorten the distance since $D_{t-1} > d(u_a,v) + d(u_a,w) \geq d(v,w)$ (see [6]). □

Lemma 7 implies that we can maintain the answer node pairs by using only the distances from the added node if the diameter grows or remains unchanged.

That is, if the added node delineates the diameter, we can compute the answer node pairs by using the distances from the added node. And if node addition shortens the distances of a previous answer node pair, that pair can be efficiently detected with from Lemma 7. If there exist no node pair whose distance is equal to the previous diameter, we detect the new diameter by the filtering algorithm.

Algorithm 2 describes our G-Scale algorithm. It first computes the maximal distance of the added node (line 1). If the maximal distance is longer than the prior time diameter, the added node must delineate the diameter (Lemma 3). It uses the distances from added node to set the diameter and the answer node pairs (lines 3-5). If the maximal distance is equal to the prior time diameter or if there exists an answer node pair whose distance is equal to the prior time diameter, the diameter remains unchanged after node addition (Lemma 5). It appends/removes the answer node pairs by distances from the added node (lines 8-13). If no node pair exists whose distance is equal to the prior time diameter after the addition, the diameter shrinks (Lemma 4). The diameter and the answer node pair are identified in Algorithm 1 (lines 15-17). Thus G-Scale limits the application of the filtering algorithm to the minimum.

Time-evolving graphs experience the addition of nodes and we assume here that a graph has only one node at $t = 1$. At $t = 1$, we set $D_{t-1} = 0$ and $\mathcal{D}_{t-1} = \emptyset$. Lifting this assumption is not difficult, and is not pursued in this paper.

Even though we assumed single node addition, we can also handle the addition of several nodes in each time tick; we simply iterate the above procedure for each additional node. If one edge is added, we assume one connected node is added to the graph. For node/edge deletion, we can detect the diameter by Algorithm 1 since such graphs do not have the property of Lemma 2. This procedure is several orders of magnitude faster than the existing approach as showed in Section 6.

We have focused on unweighted undirected graphs in this paper, but G-Scale can also handle weighted or directed graphs. For weighted graphs, we use Dijkstra's algorithm to compute distances from nodes, and bread-first search for each direction to obtain distances for directed graphs. Monitoring procedures, such as how to estimate the maximal distance from reference nodes and how to maintain the answer node pairs, are the same as those for unweighted undirected graphs.

5 Theoretical Analysis

In this section, we introduce a theoretical analysis that confirms the accuracy and complexity of G-Scale. Let k be the number of reference nodes.

5.1 Accuracy

We prove that G-Scale detects the diameter accurately (without fail) as follows:

Lemma 8 (Exact monitoring). *G-Scale guarantees the exact answer in identifying the diameter.*

Algorithm 2. G-Scale

```
Input: $G_t = (V, E)$, a time-evolving graph at time $t$
       $D_{t-1}$, the diameter at previous time tick
       $\mathcal{D}_{t-1}$, the answer node pairs at previous time tick
       $u_a$, the added node at time $t$
Output: $D_t$, the diameter of graph $G_t$
        $\mathcal{D}_t$, the answer node pairs
 1: compute the maximal distance $d_{max}(u_a)$;
 2: //Growth in diameter
 3: if $d_{max}(u_a) > D_{t-1}$ then
 4:     $D_t := d_{max}(u_a)$;
 5:     $\mathcal{D}_t := \{(u_a, v)|d(u_a, v) = D_t\}$;
 6: else
 7:     //Unchanged diameter
 8:     $D_t := D_{t-1}$;
 9:     $\mathcal{D}_t := \mathcal{D}_{t-1}$;
10:     if $d_{max}(u_a) = D_{t-1}$ then
11:         append $\{(u_a, v)|d(u_a, v) = D_{t-1}\}$ to $\mathcal{D}_t$;
12:     end if
13:     remove $\{(v, w)|d(v, u_a) + d(w, u_a) < D_{t-1}\}$ from $\mathcal{D}_t$;
14:     //Shrinkage in diameter
15:     if $\mathcal{D}_t = \emptyset$ then
16:         compute $D_t$ and $\mathcal{D}_t$ by the filtering algorithm;
17:     end if
18: end if
```

Proof. Mathematical induction can be used to prove that G-Scale detects the diameter exactly at time $t(\geq 1)$. First, we must show that the statement is true at $t = 1$. At time $t = 1$, G-Scale detects $D_1 = 0$ and $\mathcal{D}_1 = (u_1, u_1)$ exactly since (1) it sets $D_{t-1} = 0$ and $\mathcal{D}_{t-1} = \emptyset$ and (2) $d_{max}(u_1) = 0$ (see lines 8-12 in Algorithm 2). Next, we will assume that the statement holds at $t = i$. Assuming this, we must prove that the statement holds for its successor, $t = i + 1$. If the diameter does not shrink at $t = i+1$, it detects the diameter and the answer node pairs exactly from the distances from the added node with Lemma 7. Otherwise, it finds the diameter and the answer node pairs by the filtering algorithm. The filtering algorithm discards a node if its estimated maximal distance is lower than the candidate distance, and node estimation has upper bounding property (Lemma 1). That is, a node that delineates the diameter cannot be pruned. We have now fulfilled both conditions of the principle of mathematical induction. □

5.2 Complexity

We discuss the complexity of G-Scale.

Lemma 9 (Space complexity of G-Scale). *G-Scale requires $O(n+m)$ space to compute the diameter.*

Proof. G-Scale requires $O(n+m)$ space to keep the graph. The number of answer node pairs is negligible compared to that of nodes/edges as shown in Section 6. As a result, G-Scale requires $O(n+m)$ space in diameter monitoring. □

Lemma 10 (Time complexity of G-Scale). *G-Scale requires $O(n+m)$ time if the diameter does not shrink by node addition, otherwise it requires $O(kn+km)$ time to compute the diameter.*

Proof. To identify the diameter, G-Scale first compute the distances from the added node and examines whether the added node delimits the diameter or shortens the distances of the previous answer node pairs. It takes $O(n+m)$ time. If the diameter shrinks, G-Scale detects the diameter with the filtering algorithm which takes $O(kn+km)$ time. As a result it requires $O(n+m)$ time if the diameter does not shrink, and it takes $O(kn+km)$ time if the diameter shrinks. □

The monitoring cost depends on the effectiveness of the filtering and incremental update techniques used by G-Scale to detect the diameter. In the next section, we confirm the effectiveness of our approach by presenting the results of extensive experiments.

6 Experimental Evaluation

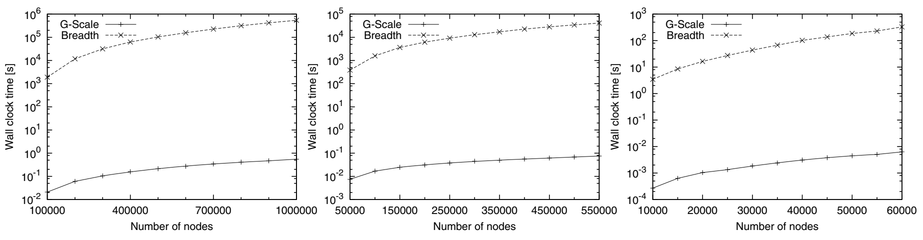
We performed experiments to demonstrate G-Scale's effectiveness. We compared G-Scale to the existing common algorithm based on breadth-first search [4] and the network structure index [17]. Note that the network structure index can compute node distances quickly at the expense of exactness. Furthermore, this method requires $O(n^2)$ space and $O(n^3)$ time as described in their paper; this method has higher orders of space and time complexities than the method based on breadth-first search.

Our experiments will demonstrate that:

- Efficiency: G-Scale outperforms breadth-first search by up to 5 orders of magnitude for the real datasets tested. G-Scale is scalable to dataset size (Section 6.1).
- Effectiveness: The components of G-Scale, reference node filtering and incremental update, are effective in monitoring the diameter (Section 6.2).
- Exactness: Unlike the existing approach, which sacrifices accuracy, G-Scale can identify the diameter exactly and efficiently (Section 6.3).

We used the following three public datasets in the experiments: *Citation*, *Web*, and *P2P*. They are a U.S. patent network, web pages within 'berkely.edu' and 'stanford.edu' domain, and the Gnutella peer-to-peer file sharing network, respectively. All data can be downloaded from [1]. We extracted the largest connected component from the real data, and we added single nodes one by one in the experiments.

We evaluated the monitoring performance mainly through wall clock time. All experiments were conducted on a Linux quad 3.33 GHz Intel Xeon server with 32GB of main memory. We implemented our algorithms using GCC.



(1) Citation (2) Web (3) P2P

Fig. 1. Wall clock time versus number of nodes

6.1 Efficiency and Scalability

We assessed the monitoring time needed for G-Scale and breadth-first search. We conducted trials with various numbers of nodes because differences in this number are expected to have strong impact for wall clock time. Figure 1 shows the wall clock time as a function of the number of nodes to detect the diameter.

These figures show that our method is much faster than breadth-first search under all the conditions examined. Breadth-first search computes distances for all node pairs in a graph. However G-Scale requires only distances from an added node if the diameter does not shrink. Even though G-Scale computes the distances from reference nodes if the diameter shrinks, this cost has no effect on the experimental results. This is because node addition hardly changes the diameter in time-evolving graphs and the number of reference nodes, k, is very small as is shown in the next section.

6.2 Effectiveness of Each Approach

In the following experiments, we examine the effectiveness of the core techniques of G-Scale: reference node filtering and incremental update.

Reference node filtering. G-Scale prunes unlikely nodes using reference nodes and the candidate distance. As mentioned in Section 4.2, G-Scale selects the highest-degree node as a reference node and utilizes the previous answer node pairs as candidate pairs. To show the effectiveness of this idea, we removed the update approach from G-Scale to directly evaluate the filtering technique, and examined the wall clock time. In other words, we directly evaluate Algorithm 1.

Figure 2 shows the result. In this figure, G-Scale without the update technique is abbreviated to *Filtering*, and *Random* represents the results where reference nodes and the candidate distance are selected at random. The numbers of nodes in this figure are $1,000,000$ for Citation, $550,000$ for Web, and $60,000$ for P2P.

Our selection methods require less computation time than the other methods. The maximal distances of the highest-degree node are expected to be short, and the prior answer node pairs are likely to remain valid. Therefore, the filtering algorithm can efficiently detect the diameter for time-evolving graphs.

For node/edge deletion, we can detect the diameter by Algorithm 1 as described in Section 4.3. Figure 2 shows the effectiveness of this approach; it is

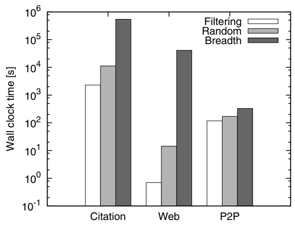
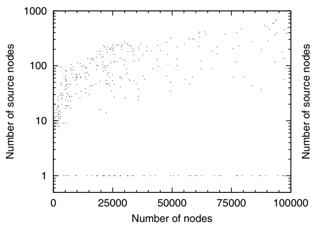
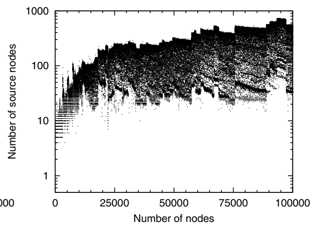


Fig. 2. Effect of reference node filtering



(1) G-Scale



(2) Filtering

Fig. 3. Number of source nodes from which distances are computed in diameter monitoring

several orders of magnitude faster than the existing approach (compare Filtering to Breadth).

Incremental update. Our update algorithm efficiently detects the diameter by reducing the application of the filtering algorithm. That is, if the diameter does not shrink, it maintains the answer node pairs by the distances from the added node. We compared the number of source nodes from which distances are computed to show the effectiveness of this approach. Note that the number of source nodes is the number of the added node plus the reference nodes. In other words, the number of source nodes is $k + 1$. Figure 3 shows the results by G-Scale and without update technique (abbreviated to *Filtering*) in monitoring time-evolving graphs. And Figure 4 shows the number of answer node pairs. We used Citation as dataset.

Figure 3 shows that the update algorithm significantly reduces the number of source nodes. As we can see from the figure, in practice, G-scale computes the distances only from the added node, while the number of reference nodes, k, is much smaller than that of graph nodes, n, with the filtering algorithm. Moreover, in most cases, the number of answer node pairs is less than 10 as shown in Figure 4; it was never more than 30 in the experiments. Therefore, it can efficiently check whether node addition invalidates the previous answer node pairs. As a result, G-scale can maintain the answer node pairs efficiently.

6.3 Exactness of the Monitoring Results

One major advantage of G-Scale is that it guarantees the exact answer, but this raises the following simple questions:

- How successful is the previous approximation approach in providing the exact answer even though it sacrifices exactness?
- Can G-Scale identify the diameter faster than the previous approximation approach that does not guarantee the exact answer?

To answer the these questions, we conducted comparative experiments using the network structure index proposed by Rattigan et al. [17]. Although they studied several estimation schemes for node distances, we compared the **distant to**

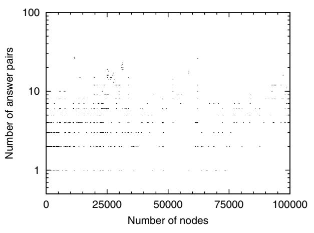
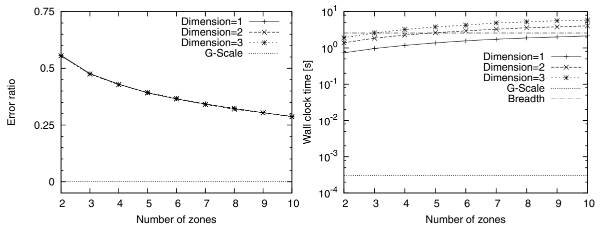


Fig. 4. Number of answer pairs



(1) Error ratio (2) Wall clock time

Fig. 5. Comparison of G-Scale and the network structure index

zone annotation scheme to G-Scale since it outperforms the other approaches, including embedding schemes mentioned in Section 2, in all of our dataset; the same result is reported in their paper. This annotation has two parameters: **zones** and **dimensions**. Zones are divided regions of the entire graph, and dimensions are sets of zones. We measured the quality of accuracy by the error ratio, which is error value of the estimated diameter distance divided the exact diameter distance. Note that the error ratio becomes a value from 0 to 1. Figure 5 shows the error ratio and the wall clock time of the diameter detection. The dataset used was Citation where the number of nodes is $10,000$.

As we can see from the figure, the error ratio of G-Scale is 0 because it identifies the diameter accurately. However, the network structure index has much higher error ratio. And the number of dimensions has no impact on the error ratio. Therefore it is not practical to use the network structure index in identifying the diameter. This answers the first question.

Figure 5 shows that G-Scale greatly reduces the computation time while it guarantees the exact answer. Specifically, G-Scale is at least $2,400$ times faster than the network structure index in this experiment; this is our answer to the second question.

The efficiency of the network structure index depends on the parameters used; it can take much more time than breadth-first search if the parameters are wrongly chosen. Furthermore, the results show that the network structure index forces a trade-off between speed and accuracy. That is, as the number of zones and dimensions decreases, the wall clock time decreases but the error ratio increases. The network structure index is an estimation technique and so can miss the exact answer. G-Scale also estimates the maximal distances to yield efficient filtering, but unlike the network structure index, G-Scale does not discard the exact answer in the monitoring process. As a result, G-Scale is superior to the network structure index in not only accuracy, but also speed.

7 Conclusions

This paper addressed the problem of detecting the diameter of time-evolving graphs efficiently. As far as we know, this is the first study to address the diameter monitoring problem for time-evolving graphs with the guarantee of exactness. Our proposal, G-Scale, is based on two ideas: (1) It filters unlikely nodes by selecting reference nodes to estimate the maximal distances, and (2) It incrementally updates the answer node pairs by exploiting the distances from the newly added node. Our experiments show that G-Scale is significantly faster than the existing methods. Diameter monitoring is fundamental for many mining applications in various domains such as content distribution, metro networks, the Internet, and citation networks. The proposed solution allows the diameter to be detected exactly and efficiently, and helps to improve the effectiveness of future data mining applications.

References

1. `http://snap.stanford.edu/data/index.html`
2. Androutsellis-Theotokis, S., Spinellis, D.: A survey of peer-to-peer content distribution technologies. ACM Comput. Surv. 36(4), 335–371 (2004)
3. Bawa, M., Cooper, B.F., Crespo, A., Daswani, N., Ganesan, P., Garcia-Molina, H., Kamvar, S.D., Marti, S., Schlosser, M.T., Sun, Q., Vinograd, P., Yang, B.: Peer-to-peer research at stanford. SIGMOD Record 32(3), 23–28 (2003)
4. Brandes, U.: A faster algorithm for betweenness centrality. Journal of Mathematical Sociology 25, 163–177 (2001)
5. Brandes, U., Erlebach, T.: Network Analysis: Methodological Foundations. Springer, Heidelberg (2008)
6. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: Introduction to Algorithms. The MIT Press, Cambridge (2009)
7. Goldberg, A.V., Harrelson, C.: Computing the shortest path: search meets graph theory. In: SODA, pp. 156–165 (2005)
8. Koppula, H.S., Puspesh, K., Ganguly, N.: Study and improvement of robustness of overlay networks (2008)
9. Kumar, A., Merugu, S., Xu, J., Yu, X.: Ulysses: A robust, low-diameter, low-latency peer-ti-peer network. In: ICNP, pp. 258–267 (2003)
10. Leskovec, J., Kleinberg, J.M., Faloutsos, C.: Graph evolution: Densification and shrinking diameters. TKDD 1(1) (2007)
11. Magoni, D., Pansiot, J.J.: Analysis of the autonomous system network topology. SIGCOMM Comput. Commun. Rev. 31(3), 26–37 (2001)
12. Newman: The structure and function of complex networks. SIREV: SIAM Review 45 (2003)
13. Ng, A.K.S., Efstathiou, J.: Structural robustness of complex networks. In: NetSci. (2006)
14. Ng, T.S.E., Zhang, H.: Predicting internet network distance with coordinates-based approaches. In: INFOCOM (2002)
15. Potamias, M., Bonchi, F., Castillo, C., Gionis, A.: Fast shortest path distance estimation in large networks. In: CIKM, pp. 867–876 (2009)
16. Ratnasamy, S., Stoica, I., Shenker, S.: Routing algorithms for dHTs: Some open questions. In: Druschel, P., Kaashoek, M.F., Rowstron, A. (eds.) IPTPS 2002. LNCS, vol. 2429, pp. 45–52. Springer, Heidelberg (2002)
17. Rattigan, M.J., Maier, M., Jensen, D.: Using structure indices for efficient approximation of network properties. In: KDD, pp. 357–366 (2006)