Net-Ray: Visualizing and Mining Billion-Scale Graphs

U Kang[1] Jay-Yoon Lee[2] Danai Koutra[2] Christos Faloutsos[2]

[1] KAIST, Daejeon 305-701, Korea
ukang@cs.kaist.ac.kr
[2] Carnegie Mellon University, Pittsburgh PA 15213, USA
{lee.jayyoon,danai,christos}@cs.cmu.edu

Abstract. How can we visualize billion-scale graphs? How to spot outliers in such graphs quickly? Visualizing graphs is the most direct way of understanding them; however, billion-scale graphs are very difficult to visualize since the amount of information overflows the resolution of a typical screen.

In this paper we propose NET-RAY, an open-source package for visualization-based mining on billion-scale graphs. NET-RAY visualizes graphs using the spy plot (adjacency matrix patterns), distribution plot, and correlation plot which involve careful node ordering and scaling. In addition, NET-RAY efficiently summarizes scatter clusters of graphs in a way that finds outliers automatically, and makes it easy to interpret them visually. To the best of our knowledge, NET-RAY is the first work for visual mining on billion-scale graphs.

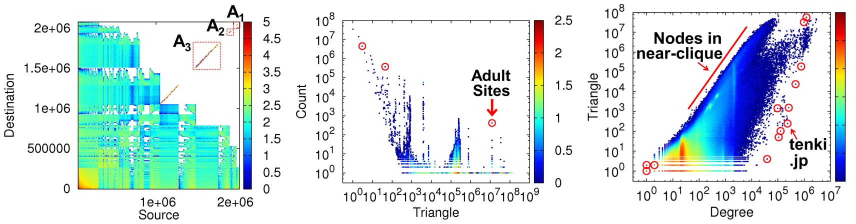
Extensive experiments show that NET-RAY handles very large graphs with billions of nodes and edges efficiently and effectively. Specifically, among the various datasets that we study, we visualize in multiple ways the YahooWeb graph which spans 1.4 billion webpages and 6.6 billion links, and the Twitter who-follows-whom graph, which consists of 62.5 million users and 1.8 billion edges. We report interesting clusters and outliers spotted and summarized by NET-RAY.

1 Introduction

Applying algorithms to find patterns in the data is one way of understanding it, but "a picture is worth a thousand words". How can we visualize big graphs with billions of nodes and edges? And, more importantly, how can we spot and plot outliers in such graphs quickly? Big graphs are everywhere: the World Wide Web, social network, biological network, phone call network, and many more. Visual mining on big graphs is a crucial tool for data miners to communicate with people outside: e.g., executives, government officials, domain experts, etc. In the case of outlier detection, visualization helps non-data miners understand the nature and seriousness of outliers.

In this paper, we propose NET-RAY, an open source package (available in `http://kdd.kaist.ac.kr/software`) implemented on top of MAPREDUCE for visual mining on big graphs. NET-RAY provides the following plots:

1. *Spy plot* (=adjacency matrix showing the nonzero patterns) of graphs, as shown in Fig. 1(a) which visualizes the adjacency matrix of US Patent graph.
2. *Distribution plot* of graph features including in/out-degrees and triangles. For example, see Fig. 1(b) for the distribution of triangles in YahooWeb graph.
3. *Correlation plot* of graph features including in-degree vs. out-degree, degree vs. triangle, and degree vs. PageRank. For instance, Fig. 1(c) shows the correlation plots between degree and Triangles of Twitter graph.



(a) Spyplot of US Patent from NET-RAY-SPY

(b) Distribution plot of YahooWeb from NET-RAY-SCATTER

(c) Correlation plot of Twitter from NET-RAY-SCATTER

Fig. 1: NET-RAY in action. **(a)** Spy plot (adjacency matrix pattern) of US Patent graph reveals communities which are labeled as A_1, A_2, and A_3. **(b)** Triangle distribution plot of YahooWeb graph reveals adult sites which are pointed by other adult sites. **(c)** Degree vs. Triangle correlation plot of Twitter graph reveals near-cliques (in the upper part of the plot), and anomalous nodes with few triangles like tenki.jp (details in Section 4).

NET-RAY uses those plots for various graph mining tasks including finding communities, discovering correlations, detecting anomalies, and visualization, as shown in Fig. 1. NET-RAY tackles two challenges. First, real world Web-scale graphs contain too much data, spanning Terabytes, for a standard single-machine plotting tools (e.g. gnuplot) to process. Moreover, the amount of information is too much to show on a standard screen with limited resolution, and thus careful reorganization and scaling of data are required. Second, even after presenting the data on the screen, finding representative outliers is difficult. NET-RAY solves the problem by distributed projection, careful ordering and scaling, and efficient summarization of representative outliers. The main contributions of this paper are the following:

1. **Method.** We propose NET-RAY, an open-source package for visualizing and mining big graphs. NET-RAY includes two algorithms: NET-RAY-SPY and NET-RAY-SCATTER. The former effectively visualizes adjacency matrices of graphs by careful ordering of nodes, and scaling values and axes. The latter efficiently finds representative outliers from big graphs.
2. **Scalability.** NET-RAY scales linearly with the number of machines and the edges in the graph.
3. **Discovery.** We employ NET-RAY to analyze large, real world data including the YahooWeb with 6.6 billion edges and total size of 0.11TB, as well as a Twitter graph with 1.8 billion edges and size of 24.2GB. We present interesting discoveries including communities and anomalous nodes. To the best of our knowledge, NET-RAY is the first work in visual mining for billion scale graphs.

The rest of the paper is organized typically: proposed methods in Sections 2 and 3, discovery results in Section 4, related works in Section 5, and conclusion in Section 6. Table 1 lists the symbols and their definitions used in this paper.

2 Proposed Method: Mining the Adjacency Matrix

Visualization of the adjacency matrix of a graph provides rich information about the connectivity patterns between the nodes, and leads to the discovery of community structures. For small graphs, visualizing the adjacency matrix is tractable. However, visualizing the adjacency matrix of very large graphs poses several challenges. First, the size

Table 1: Table of symbols.

Symbol	Definition	Symbol	Definition
n	number of nodes in a graph	m	number of edges in a graph
$\mathbf{x}, \mathbf{y}$	d-dimensional point	s	resolution (width, height) of the target matrix
k	number of clusters for NET-RAY-SCATTER	N	number of data points
$\mathbf{c}_j$	d-dimensional centroid of C_j	C_j	jth cluster

of the adjacency matrix can easily go beyond the resolution of a typical screen. For example, the adjacency matrix size of a 1 billion node graph becomes 1 billion by 1 billion; exactly visualizing the matrix requires 1 billion × 1 billion pixels which are too many to be shown on a typical screen. We address the challenge by projecting the original matrix into a small matrix which can be shown on a typical screen. For example, the 1 billion by 1 billion matrix can be projected into a 1000 by 1000 matrix, where an element of the small matrix is set to the number of nonzeros in the corresponding submatrix of the big matrix. However, this projection poses the second challenge: the small matrix will be almost full in most cases, as shown in Fig. 2 (a).

In this section, we describe our proposed NET-RAY-SPY method to address the two challenges of mining the *adjacency matrix*; the first challenge of resolution is handled in Section 2.1, and the second "full matrix" challenge is handled in Sections 2.2 and 2.3.

2.1 Projection

To handle the problem that the adjacency matrix is much larger than the screen, we project the original adjacency matrix into a small matrix which can be easily shown in a screen. In the following we assume a graph G with n nodes and m edges, and the target matrix of size s by s (e.g. $s = 1000$).

Let (x, y) be an edge in the graph, where $1 \leq x, y \leq n$. We project each edge by mapping (x, y) to the element $(\lceil x \cdot \frac{s}{n} \rceil, \lceil y \cdot \frac{s}{n} \rceil)$ in the target matrix. Note that both of the mapped values ($\lceil x \cdot \frac{s}{n} \rceil$ and $\lceil y \cdot \frac{s}{n} \rceil$) are in the range of $[1, s]$.

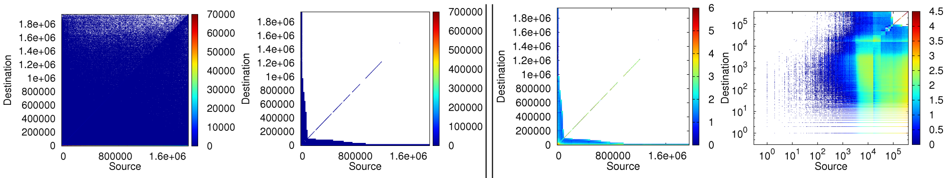
2.2 Node Reordering

Projection (Section 2.1) successfully decreases the data size, but it has a drawback: the target matrix becomes full in most cases, thereby giving a false impression that the graph is a clique or a near-clique, as in Fig. 2 (a). Identifying the communities in the graph becomes difficult in this case. To overcome this problem we propose to cluster the nonzero elements in the adjacency matrix of the original graph.

Any edge clustering method (e.g. Metis [17], CrossAssociation [6], etc.) can be plugged into NET-RAY-SPY. By default, NET-RAY-SPY uses SlashBurn [14] for clustering the nonzeros in the adjacency matrix, since it provides the best performance in terms of compression. SlashBurn reorders the nodes and assigns small node ids to high degree nodes, and clusters the nonzero elements of the adjacency matrix into the left, bottom, and diagonal area of the spy plot, thereby making huge empty areas in the spy plot. For example, see Fig. 2 (b) where SlashBurn clusters the nonzero elements of the adjacency matrix to show the connectivity patterns between the nodes.

2.3 3-LOG Scaling

In addition to the node reordering, we handle the challenge of "full matrix" by scaling the *x* and *y* axes, as well as the numerical *value* of each element into log scale.



(a) Reordering: No
Log Scaling: No

(b) Reordering: Yes
Log Scaling: No

(c) Reordering: Yes
Log Scaling: 1-LOG

(d) Reordering: Yes
Log Scaling: 3-LOG

Fig. 2: Effect of ordering and scaling in visualizing the adjacency matrix of Weibo-KDD graph. **(a,b)**: before applying NET-RAY-SPY. **(c,d)**: after applying NET-RAY-SPY. The color bar in (c,d) are in log scale of base 10. Note the node ordering and value/axis scaling provide rich information on the connectivity and activity patterns of the adjacency matrix.

We describe our proposed methods: “1-LOG” (value) and “3-LOG” (value, x, and y) scaling.

1-LOG: Value Scaling. In many real world adjacency matrices, the distribution of the nonzero elements is skewed: i.e., some areas of the adjacency matrix are very dense, while others are very sparse. For this reason, the simple linear scaling of the values loses the information on the subtle differences of the small values. For example, in Fig. 2 (a,b), most of the elements are colored blue, since few elements have the largest values ($\sim$70000) and most others values are smaller than 10% of the largest value. The red dots denoting the elements with the highest values are located near (0,0), but they are hardly visible. To resolve the problem, we use the log-scaling on the values, which we call the 1-LOG method. 1-LOG method shows the skewed distributions more effectively; e.g., in Fig. 2(c) we see that the area near (0,0) is very dense (red dots). Note the numbers along the color palette denote values raised to the power of 10 (e.g. 6 means 10^6).

3-LOG: Value and Axis Scaling. The nonzero pattern of the reordered adjacency matrix is also skewed after node reordering: most of the elements are clustered around the origin (0,0), two axes, and the diagonal line, thereby leaving many empty spaces as shown in Fig. 2(c). To better utilize the empty space, we additionally use log-scale for the two axes, and thus create the 3-LOG (value, x, and y) plot. For example, Fig. 2 (d) shows the 3-LOG plot of the Weibo-KDD graph (described in Section 4). Note the active interactions between nodes in the range of $10^4 \sim 10^5$ (source) and nodes in the range of $10 \sim 10^4$ (destination) are clearly visible.

Using log scale for the values requires carefully defining the bounding rectangle in the log-space, so that the values are properly mapped into the screen. Let x_{min} and x_{max} denote the minimum and the maximum values resp. in the x axis, after the $\log$ scaling. The values y_{min} and y_{max} are defined similarly. Our idea is to map: (a) the lower, left boundary point (x_{min}, y_{min}) to the center of the lower, left boundary pixel, and (b) the upper, right boundary point (x_{max}, y_{max}) to the center of the upper, right boundary pixel. Then remaining points (x, y) are mapped naturally to

$$(\lceil (s-1)\frac{x - x_{min}}{x_{max} - x_{min}} + \frac{1}{2}\rceil, \lceil (s-1)\frac{y - y_{min}}{y_{max} - y_{min}} + \frac{1}{2}\rceil). \quad (1)$$

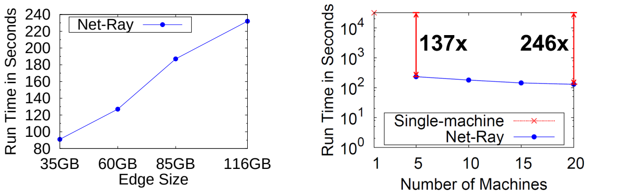
In very large graphs with billions of nodes and edges, the number of points easily exceeds billions; thus, it takes long to compute the mapping. In Section 3.1 we describe a distributed algorithm to compute the mapping.

3 Proposed Method: Mining the Scatter Cluster

In this section we describe NET-RAY-SCATTER, our proposed method for visualizing and mining *scatter plots* including distribution and correlation plots. We first describe the distributed projection method for visualizing very large scatter plots, and then the summarization/outlier detection method.

3.1 Distributed Projection

Algorithm. For very large graphs with billions of nodes and edges, using a single machine for projection using Equation (1) takes very long. To speed up the task, our natural choice is to design a distributed algorithm for the task: specifically, we use MAPREDUCE [9], a popular distributed data processing platform. We designed and implemented a two-stage MAPREDUCE algorithm for the task. Given a set $\{(x, y)\}$ of data points, the first stage finds the minimum and the maximum of each dimension: x_{min}, x_{max}, y_{min}, and y_{max}. The second stage uses the values x_{min}, x_{max}, y_{min}, and y_{max}, computed from the first stage, to find the mapping given by Eq. (1). Note that the same distributed algorithm can be used for digitizing the spy plot described in Sec. 2; in that case, the 1st MAPREDUCE stage is omitted since the minimum and maximum values are known a priori ($\log 1$ and $\log n$, resp.).



(a) Running time vs. edges. (b) Running time vs. machines.

Fig. 3: Scalability of NET-RAY on YahooWeb graph. **(a)** NET-RAY scales linearly with the edges. **(b)** NET-RAY with 5 and 20 machines is 137× and 246× faster than the single-machine counterpart, resp.

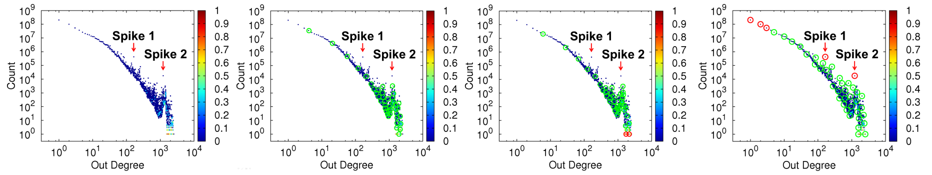
Scalability. Fig. 3 shows the scalability of NET-RAY-SCATTER on the YahooWeb graph listed in Table 2. The Hadoop-based implementation was run on the OCC-Y Hadoop cluster described in Section 4, while the single-machine implementation on a machine with two dual-core Intel Xeon 3GHz CPUs and 4GB memory. Fig. 3(a) shows the running time vs. file size, where the number of reducer machines is fixed to 5. We see that the running time scales linearly on the edges size. Fig. 3(b) presents the running time comparison between a single-machine implementation and NET-RAY-SCATTER on Hadoop. Note that NET-RAY-SCATTER with 5 and 20 machines is 137× and 246× faster than the single-machine counterpart, resp. The running time with 20 machines is $1.8\times$ (not $4\times$) faster than with 5 machines, due to overhead of running Hadoop jobs.

3.2 Summarization and Mining

The projected scatter plot still has many points: for example, the 1000 by 1000 scatter plot has at maximum 1 million points. Among these points, how can we automatically determine the representative points and outliers? We formally define the problem.

Problem 1. Given N points $\mathbf{x}_1, ..., \mathbf{x}_N$, find k representative points including top outliers (outlier score of a point x_j is the distance from x_j to its nearest neighbor). □

The choice of the representative points depends heavily on how we want to summarize the data. We list the desired properties of our summarization.



(a) Original plot.

(b) k-means annotation

(c) k-medoids annotation

(d) NET-RAY-SCATTER annotation

Fig. 4: Comparison of k-means, k-medoids and NET-RAY-SCATTER for spotting outliers in the out-degree distribution plot of YahooWeb graph. We use $k = 42$, following the choice of the parameter described in Section 3.3. Green circles denote the centroids of clusters containing more than 2 points. Red circles denote the centroids of singleton clusters. Notice that NET-RAY-SCATTER spots the two outstanding spikes of our interests, while k-means and k-medoids fail to detect them.

- **P1: Pick from Input.** The output of the summarization should be a subset of the input data points, since we want to give representative examples of the data.
- **P2: Outlier Detection.** The output of the summarization should contain outliers or extreme points so that we can use it for anomaly detection.
- **P3: Scalability.** The method should be fast and scalable.

Our proposed NET-RAY-SCATTER method uses the following main ideas: 1) use a clustering algorithm to compute k clusters where k is carefully chosen (details in Section 3.3), 2) use the center points of the k clusters as the summaries, and 3) use singleton clusters (having 1 point in their clusters) as the outliers. The main question is, which clustering algorithm should we use to satisfy the three desired properties?

Our answer is to use the k-center [10] algorithm. Given a set $\mathcal{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$ of N points, the k-center chooses a set of k points from $\mathcal{X}$ as cluster centers, $\mathbf{c}_1, \ldots, \mathbf{c}_k$ to minimize the objective function $\max_j \max_{\mathbf{x} \in C_j} ||\mathbf{x} - \mathbf{c}_j||$, where $|| \cdot ||$ denotes L_1 or L_2 norm. The effect of the $max()$ term is that an outlier, far from the rest of the points, is better to form a singleton cluster; otherwise the maximum distance between the points in the same cluster increases dramatically. Using the k-center for the summarization satisfies all the desired properties **P1**, **P2**, and **P3**. **P1** is satisfied since only the subset of the input data points are chosen. **P2** is satisfied since both singleton clusters (from outliers) and non-singleton clusters (from normal points) are spotted. Furthermore, **P3** is satisfied since the greedy version of the k-center algorithm [10] runs in $O(kN)$ time. Note that other algorithms like k-means and k-medoids do not satisfy all the properties: it can be shown that k-means violates **P1** and **P2**, and k-medoids violates **P2**.

NET-RAY-SCATTER is shown in Algorithm 1. NET-RAY-SCATTER first projects the input data using Equation (1). Then it applies the k-center algorithm with $k = \sqrt{N}$, where N is the number of data points, as we will describe in Sec. 3.3. Finally, it picks the singleton clusters as outliers, and non-singleton cluster centroids as normal points.

As an example of the outlier detection capability of NET-RAY-SCATTER, see Fig. 4 (d) whose red circles denote singleton clusters. Note that the red circles include the two outliers pointed by red arrows. Moreover, the 5 red circled points are exactly the points with top 5 outlier scores (distance from a point to its nearest neighbor), showing the effectiveness of NET-RAY-SCATTER. The effect of outliers in k-center was previously discussed in Charikar et al. [7]; however, the focus on the work is to perform 'robust'

Algorithm 1: Net-Ray-Scatter for summarization and outlier detection

```
Input: Set $\mathcal{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$ of data points.
Output: Set $\mathcal{Y}$ of outliers, and Set $\mathcal{R}$ of regular representative points from $\mathcal{X}$.
$Z \leftarrow$ project the input data using Equation (1);
$k \leftarrow \sqrt{N}$;
$C \leftarrow k$-center on $Z$;
$\mathcal{Y} \leftarrow$ singleton clusters from $C$;
$\mathcal{R} \leftarrow$ non-singleton clusters' centroids from $C$;
return $\mathcal{Y}, \mathcal{R}$;
```

clustering by ignoring outliers when building normal clusters for small k. In contrast, Net-Ray-Scatter uses sufficiently large k (details in Section 3.3) so that the outliers, which form their own clusters, are detected automatically.

3.3 Discussion

Parameter Choice. How to choose the parameter k for Net-Ray-Scatter? Obviously, k should be greater or equal to the number of outliers that we want to find. The question is, how many outliers do we want to find? Our main target for using Net-Ray-Scatter is to detect anomalous spikes in distribution or correlation plots. To detect the spike, we fix a value v in an axis (e.g., fix x coordinate), and investigate the set Y of points having the value v in the axis. If a point y in Y deviates significantly from the rest of the points in Y, then y is treated as an outlier. Based on this motivation, we choose k as the number of *distinct* coordinates in either of the axes. Assuming uniform distribution of the points, the parameter is given by $k = \sqrt{N}$.

Complexity. Our method is scalable. Specifically:

Lemma 1. Net-Ray-Scatter *runs in* $O(N^{1.5})$ *time.* □

Proof. (Sketch) The summarization step takes $O(kN)$ time with $k = \sqrt{N}$.

4 Discoveries

We present discovery results to answer the following questions.

Q1 What connectivity patterns and communities does Net-Ray-Spy reveal on real world graphs?

Q2 What patterns and anomalies does Net-Ray-Scatter detect in the distribution plots of real world graphs?

Q3 What patterns and outliers does Net-Ray-Scatter find in the correlation plots of real world graphs?

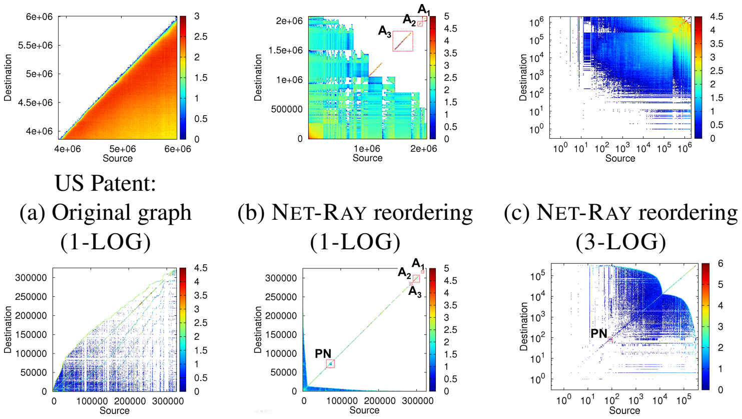
We use the graph data listed in Table 2, and for each graph we extract and analyze the following information:

- Spy plot (original, 1-LOG and 3-LOG)
- Distribution plot (in-degree, out-degree, and triangle)
- Correlation plot (in vs. out-degree, degree vs. triangle, and degree vs. PageRank)

The features (degree, PageRank, and triangles [15]) of the graphs are extracted using the Pegasus graph mining package [16]. Net-Ray is run on the OCC-Y Hadoop cluster, run by Open Cloud Consortium [1], with total 928 cores and 1 Petabyte disk.

Table 2: Order and size of networks.

Graph	Nodes	Edges	File Size
YahooWeb	1,413,511,394	6,636,600,779	116 GB
Twitter	62,539,895	1,837,645,377	24.2 GB
Weibo-KDD	1,944,589	50,655,143	594 MB
US Patent	6,009,555	10,565,431	169 MB
WWW-Barabasi	325,729	1,497,134	20 MB



US Patent:
(a) Original graph (1-LOG)
(b) NET-RAY reordering (1-LOG)
(c) NET-RAY reordering (3-LOG)

WWW-Barabasi:
(d) Original graph (1-LOG)
(e) NET-RAY reordering (1-LOG)
(f) NET-RAY reordering (3-LOG)

Fig. 5: Spy plots of real world graphs, generated from NET-RAY-SPY. The spy plot of the US Patent graph in (a) shows that patents tend to cite neither too old nor too new patents. (b) and (e) show the 'near-spoke' structures (*sparse* sub-graph loosely connected to the rest of the graph) which are labeled as A_1, A_2, and A_3. (e) and (f) also show the 'peripheral near-clique' structures (*clique-like* subgraphs loosely connected to the rest of the graph) which are labeled as 'PN'.

4.1 Spy Plots

Spy plots generated from NET-RAY-SPY provide rich information on the connectivity patterns and communities in graphs. Figs. 2 and 5 show spy plots of real world graphs. We have the following observations.

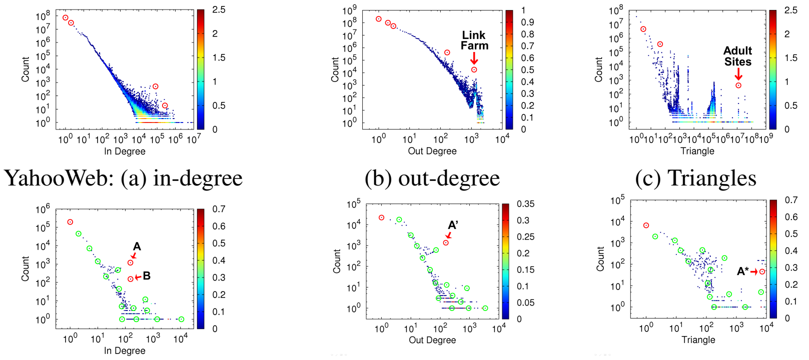
Connectivity Patterns. The spy plot enables us to easily identify the connectivity patterns in the graph. The high activity regions (yellow and red colors) of Figs. 2 (c,d), and 5 (a,b,c) show the heavy interactions between the nodes. Especially, in Fig. 5 (a) we observe that patents usually cite other patents which are neither too new nor too old: for a fixed source id, the corresponding vertical line has a single mode distribution with the maximum around the center.

Community Identification. Spy plots enable the identification of communities; especially we present sparse or dense subgraphs loosely connected to the rest of the graph.

Observation 1 ***(Near-Spokes)*** *Reordering nodes by* NET-RAY-SPY *reveals "near-spokes" (*sparse *sub-graph loosely connected to the rest of the graph).* □

For example, see Fig. 5 (b,e) for the spy plots of US Patent and WWW-Barabasi. The three squares A_1, A_2, and A_3 show the adjacency matrices of the induced subgraphs of the three near spokes.

Observation 2 ***(Peripheral Near-Cliques)*** 1-LOG *and* 3-LOG *visualization of the WWW-Barabasi graph by* NET-RAY-SPY *reveal "peripheral near-cliques"* (clique-like *subgraphs loosely connected to the rest of the graph), marked as 'PN' in Fig. 5 (e,f).* □



YahooWeb: (a) in-degree (b) out-degree (c) Triangles

WWW-Barabasi: (d) in-degree (e) out-degree (f) Triangles

Fig. 6: Distribution plots of real world graphs, generated from NET-RAY-SCATTER. First column: in-degree distribution. Second column: out-degree distribution. Third column: triangle distribution. The red circles denote the singleton clusters. The green circles in (d-f) are the non-singleton cluster centroids; we omitted green circles from (a-c) for clarity. The spikes in these degree distributions often come from anomalous activities that need attention: (b) - a link farm in the YahooWeb graph, and (d-f) - a group of nodes belonging to cliques (marked A, A', A*, and B) in WWW-Barabasi.

4.2 Distribution Plots

Distribution plots generated from NET-RAY-SCATTER provide abundant information on the regularities that graphs nodes follow, as well as deviating patterns. Fig. 6 shows the distributions of features in real world graphs.

On the regularities, notice the power law-like slopes in the distributions of degree and triangles of real world graphs. It implies the formation of links and triangles are governed mostly by "rich-get-richer" process [18].

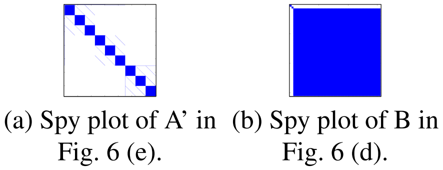
Furthermore, the distribution plots depict some "spikes" that deviate significantly from the fitting line of the majority of the nodes. We elaborate on the two types of spikes: one in the degree and the other in the triangle distributions.

Spikes In Degree Distribution. The spikes in the degree distribution often come from anomalous or special behaviors requiring attentions. The first observation is on the spike of the degree distributions in WWW-Barabasi (Fig. 6 (d,e)).

Observation 3 ***(Anomalous Spike in WWW-Barabasi)*** *Spikes in the in/out-degree distributions of WWW-Barabasi comes from cliques.* □

The spikes are observed at **A'** (in-degree 152, count 1192), **B** (in-degree 153, count 155), and **A** (out-degree 156, count 1353) of Fig. 6(d,e). It turns out that they form cliques, as we see in Fig. 7. Also, **A** is a subset of **A'**. Finally, we note that the out-degree distribution of the YahooWeb graph, shown in Fig. 6 (b), has a spike coming from a link farm [13].

Spikes in Triangle Distribution. We also observe spikes in the triangle distribution plots: **A*** in Fig. 6 (f) corresponds to a spike where all 45 nodes participate in 7239



(a) Spy plot of A' in Fig. 6 (e).

(b) Spy plot of B in Fig. 6 (d).

Fig. 7: Spy plots of the members of the spikes in distribution plots of Fig. 6(d,e). Many nodes belong to cliques; such nodes have same degree/triangle characteristics, and thus make spikes in the degree and triangle distributions.

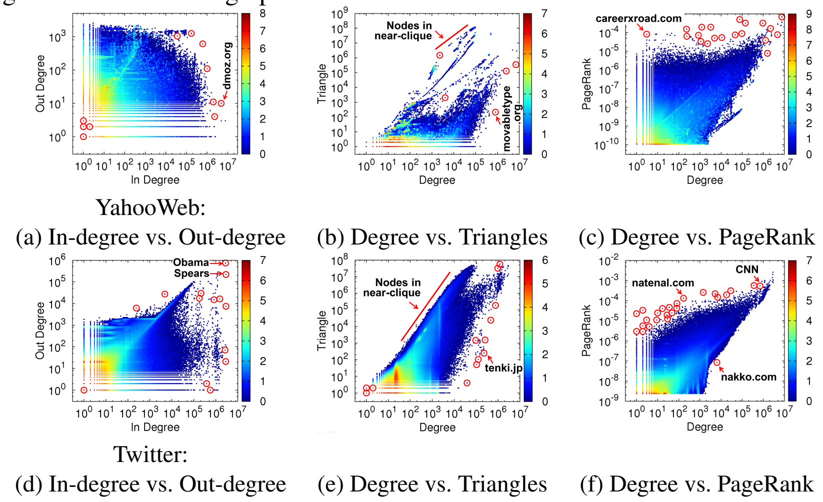
triangles. By investigating the data, we found that they form a clique, and they are subset of **A** in Fig. 6(d).

Another spike occurs in the triangle distribution of YahooWeb graph in Fig. 6 (c). The rightmost red circle contains 402 nodes having 12,420,590 triangles. Among the 402 nodes, 389 nodes have out-degrees 0 and in-degrees 81316; moreover, the incoming edges for all these 389 nodes come from the same 81316 nodes. It turns out the 389 nodes are mostly adult sites, and the 81316 sites pointing to them are other adult sites that aim at boosting the ranking of 389 nodes.

Observation 4 ***(Anomalous Spike in YahooWeb)*** *The spike in the triangle distribution of YahooWeb (Fig. 6 (c)) is due to a set of adult sites, which are pointed by other adult sites.* □

4.3 Correlation Plots

Correlation plots generated from NET-RAY-SCATTER provide opulent information on the communities, anomalous nodes, and correlation between features of nodes. Fig. 8 shows the correlation plots using the node degrees, PageRank scores, and participating triangles from real world graphs.



YahooWeb:

(a) In-degree vs. Out-degree (b) Degree vs. Triangles (c) Degree vs. PageRank

Twitter:

(d) In-degree vs. Out-degree (e) Degree vs. Triangles (f) Degree vs. PageRank

Fig. 8: Outliers in correlation plots give rich information on the structural patterns of graphs. **(Pattern 1)** Nodes with many incoming edges, but little friendship among the in-neighbors have few triangles, as shown in 'movabletype.org' of (b) and 'tenki.jp' of (e). **(Pattern 2)** On the other hand there are many nodes belonging to near-cliques, as shown in the red line of (b) and (e). **(Pattern 3)** If a node has high incoming edge vs. outgoing edge ratio, and some of the in-neighbors have high PageRank, then the node has higher PageRank than other nodes with the same degree, as shown in 'careerxroad.com' of (c), and 'natenal.com' of (f).

In and out-degrees. In Fig. 8 (a,d), popular nodes in graphs, like celebrities or portal sites, tend to have high in-degrees and small out-degrees.

Degree and Triangles. In Fig. 8(b,e) star-like nodes, which have very sparsely connected neighbors, are easily identified in the lower, right corner of the degree vs. triangle plots. Spammers can be spotted in this scheme since by their nature, the spammers often have random neighbors with very low triangles. Also near-clique communities can be spotted in the upper left corner since a node with degree d can have $\binom{d}{2} \propto d^2$ triangles at maximum.

Degree and PageRank. In Fig. 8 (c,f), higher number of in-degree, rather than total degree, is correlated with higher PageRank in general. However, it is possible to boost PageRank with small number of in degrees by having an in-neighbor with a very high PageRank. E.g, in Fig. 8 (c) a page in 'careerxroad.com' has degree 3 with high PageRank since one of the in-neighbor has a very high PageRank.

5 Related Works

Big graphs are ubiquitous and a way of sense-making is visualization. However, the existing tools cannot handle efficiently billions of nodes. Here we review some works related to visualization of graphs and outlier detection.

Graph Visualization. Apolo [8] is a graph tool for attention routing, that interactively expands the vicinities of a few seed nodes. OPAvion [2], an anomaly detection system for large graphs consists of Pegasus (feature aggregation [16]), OddBall (outlier detection [3]) and Apolo. Here, in an attempt to understand the underlying patterns and detect outliers, we are interested in efficiently generating the spy and distribution/correlation plots of a graph, instead of plotting the graph itself. In [22], Shneiderman proposes simply scaled density plots to visualize scatter plots, and [4] presents sampling-based techniques for datasets with several thousands of points.

Clusters and Outliers in Spaces. This area also comprises many works. For outliers, see LOF [5], LOCI [20], and angle-based methods [21]. For clustering, see methods like k-means in [12], G-means [11], k-harmonic means [23], k-medoids [19], and k-centers [10].

In general, there is not much work on visualization of the spy and scatter plots of features distributions and correlations for graphs with billions of nodes and edges.

6 Conclusion

In this paper, we tackle the problem of efficiently and effectively visualizing and mining billion-scale graphs. Our major contributions include:

1. **Method.** We propose NET-RAY, a carefully designed algorithm for visualizing and mining adjacency matrices and scatter plots from billion scale graphs.
2. **Scalability.** NET-RAY is linear on the number of machines and edges.
3. **Discovery.** We use NET-RAY to visualize large, real-world graphs and report interesting discoveries and anomalies found in these datasets, including near spokes, near cliques, and spikes.

To the best of our knowledge, this is the first work in visual mining for billion-scale graphs. Interesting future research directions include visual mining of dynamic graphs and complex high-dimensional data, like tensors.

Acknowledgments

Funding was provided KAIST under project number G0413002. Funding was also provided by the U.S. ARO and DARPA under Contract Number W911NF-11-C-0088, by DTRA under contract No. HDTRA1-10-1-0120, and by ARL under Cooperative Agreement Number W911NF-09-2-0053. The views and conclusions are those of the authors and should not be interpreted as representing the official policies, of the U.S. Government, or other funding parties, and no official endorsement should be inferred. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

References

1. http://opencloudconsortium.org.
2. L. Akoglu, D. H. Chau, U. Kang, D. Koutra, and C. Faloutsos. Opavion: mining and visualization in large graphs. In *SIGMOD*, 2012.
3. L. Akoglu, M. McGlohon, and C. Faloutsos. Oddball: Spotting anomalies in weighted graphs. In *PAKDD*, 2010.
4. E. Bertini and G. Santucci. By chance is not enough: Preserving relative density through non uniform sampling. In *Proceedings of the Information Visualisation*, 2004.
5. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander. Lof: Identifying density-based local outliers. In *SIGMOD*, 2000.
6. D. Chakrabarti, S. Papadimitriou, D. S. Modha, and C. Faloutsos. Fully automatic cross-associations. In *KDD*, 2004.
7. M. Charikar, S. Khuller, D. M. Mount, and G. Narasimhan. Algorithms for facility location problems with outliers. In *SODA*, 2001.
8. D. H. Chau, A. Kittur, J. I. Hong, and C. Faloutsos. Apolo: interactive large graph sense-making by combining machine learning and visualization. In *KDD*, 2011.
9. J. Dean and S. Ghemawat. Mapreduce: Simplified data processing on large clusters. In *OSDI*, pages 137–150, 2004.
10. T. F. Gonzalez. Clustering to minimize the maximum intercluster distance. *Theor. Comput. Sci.*, 38:293–306, 1985.
11. G. Hamerly and C. Elkan. Learning the k in k-means. In *In Neural Information Processing Systems*, page 2003. MIT Press, 2003.
12. J. Han, M. Kamber, and J. Pei. *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers Inc., 3rd edition, 2011.
13. U. Kang, D. H. Chau, and C. Faloutsos. Mining large graphs: Algorithms, inference, and discoveries. In *ICDE*, 2011.
14. U. Kang and C. Faloutsos. Beyond ‘caveman communities’: Hubs and spokes for graph compression and mining. In *ICDM*, 2011.
15. U. Kang, B. Meeder, E. Papalexakis, and C. Faloutsos. Heigen: Spectral analysis for billion-scale graphs. *Knowledge and Data Engineering, IEEE Transactions on*, 26(2):350–362, February 2014.
16. U. Kang, C. Tsourakakis, and C. Faloutsos. Pegasus: A peta-scale graph mining system - implementation and observations. *ICDM*, 2009.
17. G. Karypis and V. Kumar. MeTis: Unstructured Graph Partitioning and Sparse Matrix Ordering System, Version 4.0, 2009.
18. M. E. J. Newman. Power laws, pareto distributions and zipf’s law. *Contemporary Physics*, (46):323–351, 2005.
19. R. T. Ng and J. Han. Efficient and effective clustering methods for spatial data mining. In *VLDB*, 1994.

20. S. Papadimitriou, H. Kitagawa, P. B. Gibbons, and C. Faloutsos. Loci: Fast outlier detection using the local correlation integral. In *ICDE*, 2003.
21. N. Pham and R. Pagh. A near-linear time approximation algorithm for angle-based outlier detection in high-dimensional data. KDD, 2012.
22. B. Shneiderman. Extreme visualization: squeezing a billion records into a million pixels. In *SIGMOD*, 2008.
23. B. Zhang, M. Hsu, and U. Dayal. K-harmonic means - a spatial clustering algorithm with boosting. In *TSDM*, 2000.