2015 IEEE International Conference on Data Mining

Measuring Large-Scale Dynamic Graph Similarity by RICom: RWR with Intergraph Compression

Jaekoo Lee[†], Gunn Kim[♮], and Sungroh Yoon[†*]
[†]Electrical Engineering and Computer Science, Seoul National University, Seoul 151-744, Korea
[♮]Department of Physics, Sejong University, Seoul 143-747, Korea
[*]Correspondence: sryoon@snu.ac.kr

***Abstract*—‘By how much is a large-scale graph transformed over time or by a significant event?’ or ‘how structurally similar are two large-scale graphs?’ are the two questions that this paper attempts to address. The proposed method efficiently calculates and accurately produces graph similarity. Our approach is based on the well-known random walk with restart (RWR) algorithm, which quantifies relevance between nodes to express the structural and connection characteristics of graphs. Intergraph compression, which is inspired by interframe compression, merges two input graphs and reorders their nodes contributing to improved process-data storage efficiency and processing convenience. This is a boon to the RWR algorithm for large-scale graphs. The representation of a graph transformed via intergraph compression can be used to accurately show similarity because sub-matrix blocks are reordered to concentrate nonzero elements. In performing the RWR algorithm, which quantifies inter-node relevance, transformed representation of graph with intergraph compression is efficient in space requirement and produces results more quickly and accurately over conventional graph transformation schemes. We demonstrate the validity of our method through experiments and apply it to the usage data of public transportation SmartCard in a large metropolitan area to suggest usefulness of the proposed algorithm.**

I. Introduction

Graphs have been actively employed in attempts to analyze complex systems in a wide variety of applications such as traffic maps, the Internet, human brain maps, protein-protein interaction (PPI) networks [1], [2]. We pay special attention to efforts that compare and contrast two graphs by capturing their connectivity features. Research on graph similarity measurements is useful for the classification tools that exploit connectivity features extracted from static graphs, *e.g.*, a PPI network. In applications that can be abstracted with dynamic graphs, similarity in graph connectivity is monitored to detect anomalies that are manifest in the form of, for instance, accidents and intrusions [3]. Big data analysis has become an active area of research in recent years, and analyzing large-scale interconnected data such as social networking service (SNS) data is already an intensely studied topic. In particular, this paper focuses on analyzing and measuring large-scale graphs whose contents change dynamically over time or after an important event.

This paper proposes a method that efficiently measures the structural change of a dynamic large-scale graph as well as the similarity between two graphs in a large graph set. Our initial goal is to analyze a dynamic graph changing in a continuum over time. In addition, we apply our method to static graph datasets from the real world to present an optimized analysis that can offer analysis tools based on similarity.

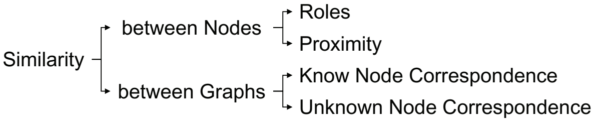

Fig. 1: Categorizing node and graph similarity measures [8]

The idea presented in this paper starts from a simple observation that perhaps some concepts in video processing could be borrowed for manipulating dynamic graphs. This is because image segmentation methods often abstract pixels into graphs and employ graph algorithms to produce meaningful results [4]. That is, there exist conceptual similarity between a video (which consists of a sequence of still images) and a dynamic graph (which is a series of adjacency matrices of static graph). As an image sequence processing method, interframe compression [5] offers improved data storage efficiency and ease of operations because it compresses common pixel data in adjacent frames. Adapting the core idea of interframe compression for graph processing, we thus propose an intergraph compression method, which compresses adjacent static graphs in a dynamic graph by exploiting the shared structure features between graphs. The proposed method can measure changes in dynamic graphs with the RWR algorithm [4] via the Schur complement [6], [7] for efficiency and accuracy.

In our experiments, the proposed method indeed produced accurate results, and its processing time and storage requirement were also improved over existing state-of-the-art methods. The validity and fitness of the proposed algorithm were experimentally tested using synthetic graph data [3] and real-world graph data [9]. The method was also tested with the dynamically changing subway SmartCard data extracted from a large metropolitan public transportation system.

The rest of the paper is organized as follows: Section 2 provides background materials. Section 3 presents the details of our approach and compares it with existing state-of-the-art methods. Section 4 shows our experimental results with both synthetic and real-word datasets and also analyzes the time and space complexity of the proposed method. Section 5 concludes the paper. The descriptions of the symbols used in this paper are listed in Table I.

II. Preliminaries

A. *Graph Similarity Measure*

As smart devices have become widespread, social network services such as Facebook or Twitter generate large-scale

1550-4786/15 $31.00 © 2015 IEEE
DOI 10.1109/ICDM.2015.123

TABLE I: The symbols and descriptions used in this paper

symbol	description
G or $G(V,E)$	graph, graph with *V*: set of node, *E*: set of edge
G_U or $G_U(V_U,E_U)$	union graph of graphs G_1 and G_2
n, m	number of nodes, number of edges
c	probability of restarting
k	number of hubs removed in node reordering method
w	number of total hubs, $w \ll n$
$sim(G_1,G_2)$	similarity between graphs G_1 and G_2
$\mathbf{I}$	$[n \times n]$ identity matrix
$\mathbf{A}$	$[n \times n]$ adjacency matrix of graph G
$\mathbf{A}'$	$[n \times n]$ inter-graph compressed adjacency matrix of $\mathbf{A}$
$\mathbf{D}$	$[n \times n]$ diagonal degree matrix of graph G, $d_{ii} = \sum_j a_{ij} (d_{ij} = 0\ for\ i \neq j)$
$\mathbf{D}'$	$[n \times n]$ inter-graph compressed diagonal degree matrix of $\mathbf{D}$
$\mathbf{B}_{ij}$	$[n_i \times n_j]$ $(i,j)^{th}$ block matrix of $\mathbf{B}$
$\mathbf{P}$	$[n \times n]$ matrix $\mathbf{P} = [\mathbf{I} - (1-c)\mathbf{A}\mathbf{D}^{-1}]$
$\mathbf{P}'$	$[n \times n]$ matrix $\mathbf{P}' = [\mathbf{I} - (1-c)\mathbf{A}'\mathbf{D}'^{-1}]$ $\mathbf{P}' = [\mathbf{P}'_{11}, \mathbf{P}'_{12}; \mathbf{P}'_{21}, \mathbf{P}'_{22}]$ 2 by 2 block partitioned matrix
$\vec{e_i}^*$	$[n \times 1]$ starting indicator (seed) vector
$\vec{r_i}$	$[n \times 1]$ relevance vector of node i
r_{ij}	relevance value of node j w.r.t. node i
$\mathbf{R}$	$[n \times n]$ matrix with elements r_{ij} $\mathbf{R} = [\mathbf{R}_{11}, \mathbf{R}_{12}; \mathbf{R}_{21}, \mathbf{R}_{22}]$ 2 by 2 block partitioned matrix
$\mathbf{S}'$	$[w \times w]$ Schur complement of block $\mathbf{P}'_{11}$ of matrix $\mathbf{P}'$

* unit vector with 1 in the i^{th} element.

graph data, which has been the focus of active research. For example, the connectivity of networks can be expressed in a dynamically changing graph for analysis, or individuals can be grouped according to the brain connectivity graph [3]. In one study, data was collected from the functional parts of the human brain and converted into a graph for analysis to classify people. Such studies measure similarity between graphs in terms of structure and/or features and allow for intuitive comparisons [8]. This paper also attempts to capture and numerically represent graph similarity. We provide an overview of the general characteristics of conventional graph measures that pertain to our work. Fig. 1 shows that our proposed method belongs to the category of graph research that excludes unknown node correspondence because it compares graphs based on approximated information of the structural role of node and connectivity features. So, it applies to graph research where node correspondence is assumed.

State-of-the-art graph similarity measures satisfy the following properties [3]. Similarity measure needs to reflect *Edge Importance*, which states that insertion and removal of edges between graphs need to be accounted for. *Weight Awareness* also needs to be satisfied so that any change to the weight of each edge needs to be reflected. Similarity measure also needs to be sensitive to *Edge Sub-modularity*, hence changes to higher-degree nodes need to have less impact than changes to lower-degree nodes. Assuming that changes in graphs due to accidents are more frequent than arbitrary changes in edges, the sensitivity measure needs to identify edge changes surrounding specific nodes, satisfying *Focus Awareness*. Through experiments, we verify that our method satisfies the four properties above.

B. Random Walk with Restart (RWR)

The RWR algorithm attempts to measure the affinity or relevance of two nodes in a graph [4]. Conceptually, in RWR, a particle walks between nodes following randomly chosen edges. The particle starts walking in the initial node i and chooses the next node connected through the edges, each of which is assigned weighted probability, and then makes its move. Repeating moves in this fashion multiple times allows for estimation of the probability of each node having the particle on it. This is a quantified value tailored by the structure of graph connectivity, hence is used as a relevance measure between two nodes [10]. The relevance score produced by the RWR algorithm accounts for the entire connectivity structure of the graph, so is employed as an essential analysis tool.

Since its introduction, the RWR algorithm has developed to accommodate expanding graphs and to numerically measure the relevance of nodes in such graphs while improving on time complexity and accuracy [4], [10], [11]. Variations of the RWR algorithm have been employed in earlier applications such as automatic image captioning [4], and are expanding into various fields. Research on the RWR algorithm encompasses practical applications, optimization for computation, storage, and noticeably for large-scale graphs as theoretical backgrounds. Some of the well-known variations of the RWR algorithm include PageRank, electrical network analogy, and personalized RWR [10]. Related studies have been applied to discover user similarity or influential individuals in SNS [12].

The iterative form of a basic RWR algorithm [4], [10] is:

$$\vec{r_i}^{(t)} \leftarrow (1-c)\mathbf{A}\mathbf{D}^{-1}\vec{r_i}^{(t-1)} + c\vec{e_i} \tag{1}$$

where c is the probability of restarting the random walk from the initial node, $\vec{e_i}$ is the starting indicator vector, $\vec{r_i}$ is the unknown relevance (affinity or influence) column vector, and (t) is the number of renewal activities for $\vec{r_i}^{(t)}$. The vector $\vec{r_i}^{(t)}$ is obtained by repeatedly processing Eq. 1 while $|\vec{r}^{(t)} - \vec{r}^{(t-1)}| < threshold$ holds true. The component r_{ij} in the produced vector $\vec{r_i}^{(t)}$ is the relevance of node *j* w.r.t. node *i*. In other words, r_{ij} is the numerically translated value of influence by node *i* for node *j* in the context of a connectivity structure. Intuitively, node *i* has more influence on node *j* if the edge linking node *i* to node *j* is numerous, short, or heavily weighted. The RWR algorithm in an iterative method repeatedly performs multiplication operations for an $[n \times n]$ matrix in each iteration to compute $\vec{r_i}^{(t)}$, hence is computationally inefficient.

When $0 < c < 1$, $\vec{r^{(t)}}$ generally approaches a unique solution [10]. Thus, the RWR algorithm can be defined as the following because $\vec{r}$ has a solution

$$[\mathbf{I} - (1-c)\mathbf{A}\mathbf{D}^{-1}]\vec{r_i} = c\vec{e_i}. \tag{2}$$

With this equation, $\vec{r_i}$ can be rewritten by means of an inverse matrix operation in linear algebra as follows:

$$\vec{r_i} = c[\mathbf{I} - (1-c)\mathbf{A}\mathbf{D}^{-1}]^{-1}\vec{e_i} = c\mathbf{P}^{-1}\vec{e_i}. \tag{3}$$

This RWR algorithm suffers from the fact that an inverse matrix operation has to be performed for an $[n \times n]$ matrix to obtain $\vec{r_i}$, and that its computational complexity $O(n^3)$ increases as n becomes larger [6]. There are several studies that have attempted to reduce time complexity, some of which are listed in Table II. As charted in the table, variations of

TABLE II: Related work on the RWR approach

methods	details
Basic Iteration [10]	$\vec{r}^{(t)} \leftarrow (1-c)\mathbf{A}\mathbf{D}^{-1}\vec{r}^{(t-1)} + c\vec{e}$
Inversion [10]	$\vec{r} = c[I - (1-c)\mathbf{A}\mathbf{D}^{-1}]^{-1}\vec{e}$
QR Decomposition [11]	$\vec{r} = c\mathbf{R}^{-1}\mathbf{Q}^T\vec{e}$
LU Decomposition [3]	$\vec{r} = c\mathbf{U}^{-1}\mathbf{L}^{-1}\vec{e}$
B_LIN [4]	$\vec{r} = c[\mathbf{I} - (1-c)\mathbf{A}_1 - (1-c)\mathbf{A}_2]^{-1}\vec{e}$ $\approx c[\mathbf{I} - (1-c)\mathbf{A}_1 - (1-c)\mathbf{U}\Sigma\mathbf{V}]^{-1}\vec{e}$
Fast Belief Propagation [3]	$\vec{r} = c[\mathbf{I} - \epsilon^2\mathbf{D} - \epsilon\mathbf{A}]^{-1}\vec{e}$
BEAR [7]	$\vec{r} = [r_1; r_2]$ where $r_1 = \mathbf{H}_{11}^{-1}(cq_1 - \mathbf{H}_{12}r_2)$ and $r_2 = \mathbf{S}^{-1}(cq_2 - \mathbf{H}_{21}\mathbf{H}_{11}^{-1}(cq_1))$
Proposed	$\vec{r} = c[I - (1-c)\mathbf{A}'\mathbf{D}'^{-1}]^{-1}\vec{e}$

$\mathbf{P} = \mathbf{QR}$, $\mathbf{Q}^{-1} = \mathbf{Q}^T$; $\mathbf{P} = \mathbf{LU}$; divide $\mathbf{AD}^{-1}$ into inner community edges $\mathbf{A}_1$ and cross community edges $\mathbf{A}_2$, $\mathbf{A}_2 \approx \mathbf{U}\Sigma\mathbf{V}$ (low-rank matrix); $\mathbf{H}$ is the partitioned matrix from Slashburn, and $\mathbf{S} = \mathbf{H}_{22} - \mathbf{H}_{21}\mathbf{H}_{11}^{-1}\mathbf{H}_{12}$; $\epsilon = 1/(1 + \max_i \mathtt{d}_{ii})$, positive constant ($< 1$) encoding neighbor influence.

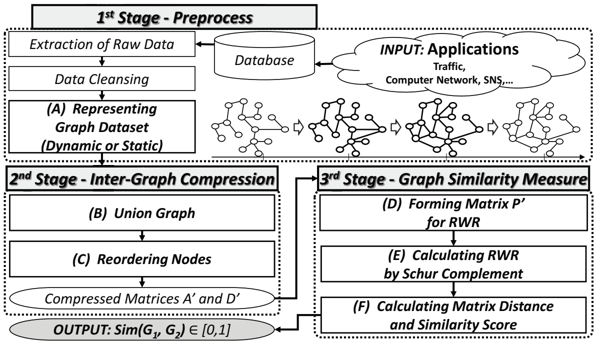
1st Stage - Preprocess
Extraction of Raw Data
Database
INPUT: Applications
Traffic,
Computer Network, SNS,...
Data Cleansing
(A) Representing Graph Dataset (Dynamic or Static)
2nd Stage - Inter-Graph Compression
(B) Union Graph
(C) Reordering Nodes
Compressed Matrices A' and D'
3rd Stage - Graph Similarity Measure
(D) Forming Matrix P' for RWR
(E) Calculating RWR by Schur Complement
(F) Calculating Matrix Distance and Similarity Score
OUTPUT: Sim(G1, G2) ∈ [0,1]

Fig. 2: Overview of proposed method

RWR include iterative methods, inversion methods [10], QR decomposition methods [11], RPPR/BRPPR methods [10], B_LIN/NB_LIN methods [4], fast belief propagation-based (DeltaCon) methods [3], and LU decomposition methods [7]. The RWR algorithm to measure the structural feature of graph for our method shows improved time/space efficiency over state-of-the-art RWR variations, and it has been devised to fit for large-scale graphs.

C. Schur Complement

The Schur complement [6] is applied as a part of the solution to a linear equation system, and is an important tool for numerical analysis and matrix operation. In graph mining, its use was pioneered by the block elimination approach for RWR (BEAR) [7]. In the proposed method, it is also an essential tool because it allows the RWR algorithm to perform efficiently. In case the size of a matrix $\mathbf{M} = [\mathbf{W}, \mathbf{X}; \mathbf{Y}, \mathbf{Z}]$ is $[(p+q) \times (p+q)]$, then its constituent block sub-matrices **W**, **X**, **Y**, and **Z** have the sizes of $[p \times p]$, $[p \times q]$, $[q \times p]$, $[q \times q]$, respectively, where $p \ll q$ in general. If **Z** is invertible, then the Schur complement for block **Z** in matrix **M** can be defined as $\mathbf{S} = \mathbf{W} – \mathbf{X}\mathbf{Z}^{-1}\mathbf{Y}$, and can be used to calculate the inverse matrix of matrix **M** as follows:

$$\mathbf{M}^{-1} = \begin{bmatrix} \mathbf{W} & \mathbf{X} \\ \mathbf{Y} & \mathbf{Z} \end{bmatrix}^{-1} = \begin{bmatrix} \mathbf{S}^{-1} & -\mathbf{S}^{-1}\mathbf{X}\mathbf{Z}^{-1} \\ -\mathbf{Z}^{-1}\mathbf{Y}\mathbf{S}^{-1} & \mathbf{Z}^{-1}+\mathbf{Z}^{-1}\mathbf{Y}\mathbf{S}^{-1}\mathbf{X}\mathbf{Z}^{-1} \end{bmatrix}.$$

D. Measurement of Matrix Distance

There have been various attempts to find similarity or distance between matrices, such as using matrix correlation or cosine similarity [6]. In this paper, we choose Euclidean distance because it allows for finding distance, which represents the relevance of each node in the graph as calculated with the RWR algorithm; the similarity between two graphs is intuitively represented.

III. Proposed Method

We present a method to efficiently measure (1) the change of a dynamic graph whose connectivity structure evolves over time and (2) the similarity of large-scale graph datasets along with justification for our method. The definition of the problem we want to address can be stated briefly as follows [3]:

Given: two graphs $G_1(V_1, E_1)$ and $G_2(V_2, E_2)$ each consisting of node set *V* and edge set *E*, we assume that the node correspondence of sets V_1 and V_2 is known.

Find: $sim(G_1, G_2)$ that lies between 0 and 1 and intuitively signifies the structural similarity between two given graphs. Here, $sim(G_1, G_2) = 0$ indicates that two graphs are structurally complementary, where $sim(G_1, G_2) = 1$ means two graphs are completely identical.

Fig. 2 summarizes the overall architecture of our method. In the first stage of our method, the provided dataset is converted to a dynamic graph, and the two adjacent static graphs $G_1(V_1, E_1)$ and $G_2(V_2, E_2)$ extracted from the given dynamic graph go through the following steps in sequence. Since intergraph compression borrows its core concept from interframe compression, the first task is creating the union set G_U with the two input graphs so that the commonly shared structural features can be relocated for compression. Afterwards, a state-of-the-art node reordering algorithm is applied to G_U, which stores common edges. In the adjacency matrix, nonzero elements indicate edges between nodes. The nonzero elements are concentrated to a focused region. This not only brings in a compressing effect on the matrix, but also facilitates partitioning in applying the RWR algorithm and provides a basis for applying the Schur complement algorithm.

Following the reordered sequence computed in intergraph compression, each graph is also reordered in the same node index, and matrix $\mathbf{P}'$ is calculated to apply the RWR algorithm so that the connectivity between nodes can be identified. To $\mathbf{P}'$, a 2-by-2 block partitioned matrix, Schur complement algorithm is applied and the relevance matrix **R** is calculated, which shows the affinity between nodes in the structural context of the entire graph. The relevance matrices $\mathbf{R}_1$ and $\mathbf{R}_2$ are efficiently calculated with a partitioned inverse matrix operation through the Schur complement algorithm. Both $\mathbf{R}_1$ and $\mathbf{R}_2$ share an identical node index and are evaluated to numeric values following matrix distance measuring operations between corresponding block components due to the partitioning block properties. The value, which lies between 0 and 1, intuitively expresses similarity between two graphs, and is the final result of our method. The numeric results obtained this way can be used as a tool to measure changes in a dynamic graph or to numerically express similarity between graphs.

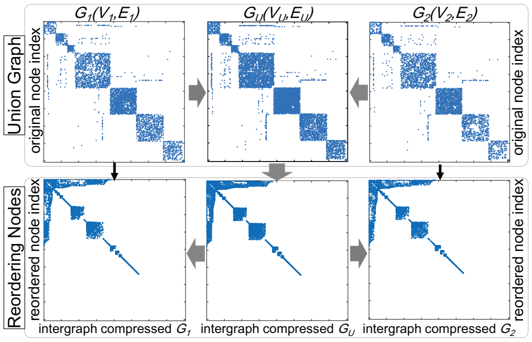

Fig. 3: Visualization of intergraph compression on real-world graphs: Union graph is an example of the merged graph of $G_U(V,E) = G_1(V_1,E_1) \cup G_2(V_2,E_2)$ for intergraph compression, and this is applicable when every single node in V_1 of $G_1(V_1,E_1)$ has its corresponding node in V_2 of $G_2(V_2,E_2)$ as well as when $(V_1 \neq V_2)$ because the node set of G_U can be defined as $V = V_1 \cup V_2$. In reordering nodes, SlashBurn removes hubs with the highest degree and the edges connected to the removed hubs, and produces spokes with smaller degrees. The hub nodes are relocated to the front of the index and the spokes to the back repeatedly to reorder the nodes. As a result, the hubs with higher degrees in the adjacency matrix for the given graph, are concentrated closely together, producing compression and partition effects. With intergraph compression, it is possible to verify shared structural data between two graphs by merging the target graphs. Node reordering algorithm is applied on the merged graph to get the compression effect like in motion picture processing, hence improving space and computation efficiency.

A. *Representing Graph Dataset*

Several methods have been proposed to visually analyze the interactive relation of data components in a given dataset using graphs [1]. In the preprocessing stage of our method, each piece of data is converted to a node and the relation between data to edge to represent input data as a graph, which is one of the simplest approaches of graph conversion. This approach is valid in converting not only static datasets but also dynamic datasets, which change over time or after an event. To the generated graph, our method is applied for clustering (static datasets) or for tracing changes (dynamic datasets).

B. *Union Graph*

The proposed method is inspired from an observation that commonly shared data between adjacent still frames of motion pictures can be processed in compressed form. The same approach may be applicable to two adjacent graphs in a dynamic graph setting. Note that for a dynamic graph in the real world [9], changes almost always happen in a continuum, and abrupt changes virtually never arise. With this in mind, the concept of compressibility for shared pixel data between adjacent frames using interframe compression for motion pictures is applied to two dynamically changing graphs, capturing shared structural features between two input graphs. Our method is comparable to interframe compression in offering storage efficiency and ease of operations.

C. *Reordering Nodes*

Several node reordering methods are available for use: degree sort, cross association, spectral clustering, and shingle, all of which bring in compressing effects [13]. For our approach, we note that most real graphs possess the pattern of a scale-free network. Inspired by BEAR [7], we thus choose SlashBurn [13], a state-of-the-art node reordering algorithm that concentrates nonzero components in adjacent matrices as much as possible, producing an excellent compression effect. Fig. 3 clearly shows the compression effect of intergraph compression after applying this algorithm to G_U, which is the union of the two given graphs.

D. *Forming Matrix* $\mathbf{P}'$ *for RWR*

We show that block-form sub-matrices belonging to one large-scale matrix are created after compressing two input graphs with intergraph compression. Let matrix $\mathbf{A}'$ be the output of intergraph compression. We apply the RWR algorithm [4] to matrix $\mathbf{A}'$ to identify inter-node connections and find matrix $\mathbf{R}$, which consists of r_{ij} (inter-node relevance).

For an arbitrarily selected node i, $\vec{r}_i$ can be expressed as

$$\vec{r}_i = c[\mathbf{I} - (1-c)\mathbf{A}'\mathbf{D}'^{-1}]^{-1}\vec{e}_i = c\mathbf{P}'^{-1}\vec{e}_i. \qquad (4)$$

After connecting each $\vec{r}_i$ of node i contiguously $(0 < i < n)$, we can calculate matrix $\mathbf{R}$ for all of the nodes in the graph that has each $\vec{r}_i$ for its column vector [3] as follows:

$$\mathbf{R} = \left[\ \vec{r}_1 \quad \ldots \quad \vec{r}_n\ \right] = \mathbf{P}'^{-1}\left[\ \vec{e}_1 \quad \ldots \quad \vec{e}_n\ \right] = \mathbf{P}'^{-1}\mathbf{I} = \mathbf{P}'^{-1}.$$

The matrix $\mathbf{P}' = [\mathbf{I} - (1-c)\mathbf{A}'\mathbf{D}'^{-1}]$ is obtained from reordered adjacency matrix $\mathbf{A}'$. Thus, the blocks are similar, and can be expressed with 2-by-2 block matrices as in matrix $\mathbf{A}'$.

With the compression effect present in $\mathbf{A}'$ due to intergraph compression, $\mathbf{P}'$ is partitioned into sub-matrices of $\mathbf{P}'_{11}$, $\mathbf{P}'_{12}$, $\mathbf{P}'_{21}$, and $\mathbf{P}'_{22}$. $\mathbf{P}'_{11}$ of $\mathbf{P}'$ is a smaller partition ($[w \times w]$) with absolutely high concentration of nonzero components, whereas the two blocks of $\mathbf{P}'_{12}$ ($[w \times (n-w)]$) and $\mathbf{P}'_{21}$ ($[(n-w) \times w]$) are symmetrical, like a pair of wings; the nonzero components are thus appropriately scattered depending on the connectivity of the graph. $\mathbf{P}'_{22}$, which takes most of the region with ($[(n-w) \times (n-w)]$) in the $\mathbf{P}'$ matrix, has only a few nonzero components to form a sparse nonzero block diagonal. Such a block diagonal sub-matrix is invertible because each sparse block is partitioned and approximated with that goal.

As the small sparse block matrix diagonal to matrix $\mathbf{P}'_{22}$ has a relatively much smaller size compared to the corresponding sub-matrix in the original matrix, each block is treated as an independent matrix, and the inverse matrix of the sparse matrix is obtained. The inverse matrix of the sparse block diagonal matrix is then calculated with approximation reflecting the location of each block. Matrix $\mathbf{P}'$ is a 2-by-2 block matrix. To find $\mathbf{R}$, which signifies node relevance as a matrix, the inverse matrix of $\mathbf{P}'$ is obtained with the Schur complement algorithm [6], [7]. This offers the benefit of partitioning a large-scale graph matrix into blocks for improved storage efficiency.

E. *Calculating RWR Matrix by Schur Complement*

Calculating the inverse of matrix $\mathbf{P}'$ for each graph with the RWR algorithm allows for finding $\mathbf{R}$, which numerically indicates inter-node relevance in the context of overall graph structure. Exploiting the techniques used in BEAR [7], the

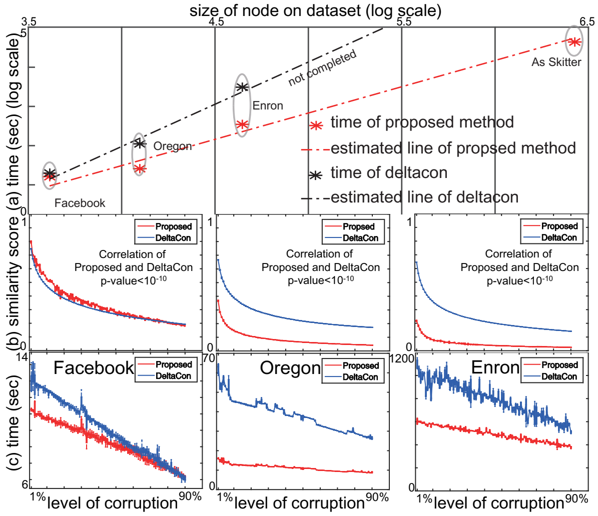

Fig. 4: Comparison of the proposed and existing approaches: (a) average processing time; (b) shows that the proposed method has faster performance in most cases over state-of-the-art DC [3], which uses approximations to calculate relevance with fast belief propagation for the RWR algorithm. This performance gap is increasingly significant for larger graphs. In the experiment to measure similarity with varying levels of corruption as shown in (b) and (c), fluctuation in the performance speed of DC for arbitrary insertion/removal of edges is increasingly noticeable for larger graphs whereas the proposed method has a relatively stable performance speed. In the similarity measurement experiment, DC and our method show exponentially decreasing similarity in the same fashion, while the number of arbitrarily altered edges increased. In our experiment, the ratio of changed insertion/deletion edges varies from 1% to 90%. When edge connection is altered hundreds of times at each ratio, DC is insensitive to arbitrary changes to edges and thus produces values in a narrow range. In contrast, similarity values produced by our method are sensitive to random insertion/removal of edges, reflecting structural changes in the graph; Real datasets [9]-Facebook (n=4039, m=88234, social network), Oregon (n=11461, m=32730, autonomous systems graph), Enron (n=36692, m=183831, communication network), AS Skitter (n=1696415, m=11095298, Internet topology graph)

steps for obtaining an inverse matrix using the Schur complement algorithm is applied to $\mathbf{P}'$ of each graph to find matrix $\mathbf{R} = c\mathbf{P}'^{-1}$, which is given by

$$\mathbf{R} = c\begin{bmatrix} \mathbf{S}'^{-1} & -\mathbf{S}'^{-1}\mathbf{P}'_{12}\mathbf{P}'^{-1}_{22} \\ -\mathbf{P}'^{-1}_{22}\mathbf{P}'_{21}\mathbf{S}'^{-1} & \mathbf{P}'^{-1}_{22} + \mathbf{P}'^{-1}_{22}\mathbf{P}'_{21}\mathbf{S}'^{-1}\mathbf{P}'_{12}\mathbf{P}'^{-1}_{22} \end{bmatrix}.$$

The resulting matrix $\mathbf{R}$ has more accurate values over the existing approximation of RWR [3] in terms of inter-node relevance. The computation also finishes more quickly while requiring less space.

F. Calculating Matrix Distance and Similarity Score

In the previous subsection, matrix $\mathbf{R}$, which numerically indicates inter-node relevance in the context of overall graph structure, was obtained from a block-partitioned matrix $\mathbf{P}'$ by means of the Schur complement. Using a distance-measuring method between matrices $\mathbf{R}_1$ and $\mathbf{R}_2$ [6], the similarity between two graphs is calculated and numerically represented:

$$sim(G_1, G_2) = 1/(\sum_{i=1}^{2}\sum_{j=1}^{2} ED(\mathbf{R}_{1,ij}, \mathbf{R}_{2,ij}) + 1). \quad (5)$$

Note that graph similarity reflecting overall graph structure ends up lying in between 0 and 1, as governed by Eq. 5:

TABLE III: Details of real-world datasets used [9]

properties	RWR	ours	DC	VEO	SS	GED	λ-D A	λ-D L	λ-D NL
edge importance	O	O	O	X	X	X	X	X	X
weight awareness	O	O	O	X	O	X	O	O	O
edge sub-modularity	O	O	O	O	O	O	O	O	O
focus awareness	O	O	O	-	-	-	-	-	-

Similarity measures [3]: DeltaCon (DC); Vertex/Edge Overlap (VEO); Signature Similarity (SS); Graph Edit Distance (GED); λ-D Adjacency (λ-D A); λ-D Laplacian (λ-D L); λ-D Normalized Laplacian (λ-D NL).

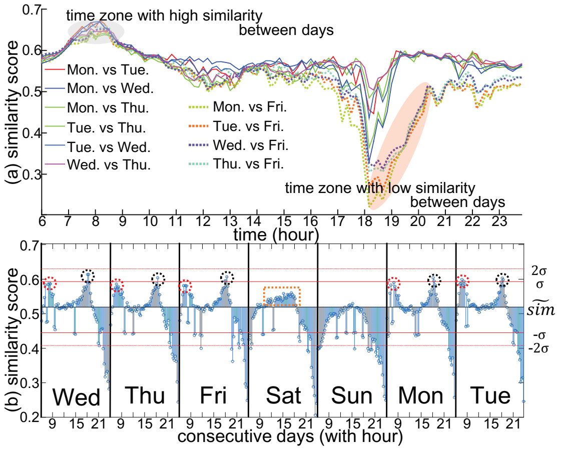

Fig. 5: Analysis of “SmartCard big data” by our approach: (a) Comparison similarity of a day of week is the result for the static graph analysis, and it depicts similarities between the flows for different days of the week. From Mon to Fri, morning rush hours between 07 and 09 have more similar passenger flows, but the passenger flow pattern on Fri evening between 18 and 20 was noticeably different from that on the other weekdays, signifying that passengers on Fri go to areas that are different from where they usually go on the other weekdays. Manually tracing passenger flow pattern on Fri verified that a significant portion of passengers do not return to the subway station from which they first left in the morning and instead go to stations near the downtown area, probably to enjoy Fri evening; (b) Monitoring pattern of passenger movement on consecutive time shows the similarity of passenger flow for each time slot in a dynamic graph. Changes in the graph are depicted in a control chart, with its mid-line and $\pm\sigma$, $\pm2\sigma$. The similarity of graphs increases during morning rush hours on weekdays until most commuters seem to have arrived at work. The graphs start to deviate from this point. The graphs start to look different after the time when most commuters are assumed to have arrived at work. In the afternoon after 15, the graphs become similar until the peak at 18. As the flow of passengers increases around 18 when commuters start returning home, the hourly graphs exhibited the highest similarity at that time. Late at night after 22, the number of passengers drops, and passenger flow is irregular, so that the transportation graphs differ very much from each other. Unlike the passenger flow patterns during weekdays, the passenger flow does not change much during the daytime on Sat. Throughout Sun, the passenger flow pattern shows less similarity to the other days while being mixed with some irregularity.

IV. Results and Discussion

We consider the four principled properties of graph similarity used in evaluating state-of-the-art methods [3]: edge importance, weight awareness, edge-submodularity, and focus-awareness. Existing measures for graph similarity listed in Table III and our method are given the same simple synthetic graph as previous work [3]. We chart similarity measure results for each method and show that our method fulfills the four properties. Table III compares the experimental results of existing methods with ours in terms of the principled properties on graph similarity. Fig. 4 presents the result from

TABLE IV: Complexity analysis of proposed approach

	action	time	space
Input: G_1 and G_2 from dynamic dataset, c probability of restarting **Output:** *sim*$(G_1, G_2) \in [0, 1]$			
(A)	represent graph set from raw dataset		
(B)	union graph	$O(m_1 + m_2) \approx O(2m)$	$O(\lvert m_1 \cup m_2 \rvert)$
(C)	reordering nodes	$O(m + n \log n \lvert T \rvert)$ [13]	$O(n)$ [13]
(D)	forming matrix $\mathbf{P}'$ for RWR	$O(mn + n)$	$O(m + n)$
(E)	calculating RWR by Schur Complement		
(E-1)	compute $\mathbf{P}_{22}'^{-1}$	$O(\sum_{l=1}^{q} n_{(l)}^3)$	$O(\sum_{l=1}^{q} n_{(l)}^2)$
(E-2)	compute $\mathbf{S}'$	$O(w^2 + n^2(\sum_{l=1}^{q} n_{(l)}^2))$	$O(w^2)$
(E-3)	compute $\mathbf{R}_{11}$	$O(w^3)$	$O(w^2)$
	compute $\mathbf{R}_{12}$	$O(w^2 n(\sum_{l=1}^{q} n_{(l)}^2))$	$O(nw)$
	compute $\mathbf{R}_{21}$	$O((\sum_{l=1}^{q} n_{(l)}^2)nw^2)$	$O(nw)$
	compute $\mathbf{R}_{22}$	$O(\sum_{l=1}^{q} n_{(l)}^2 +$ $(\sum_{l=1}^{q} n_{(l)}^2)n^2w^2(\sum_{l=1}^{q} n_{(l)}^2)$	$O((n-w)^2)$
(F)	calculating similarity score	$O(n^2 + 1)$	$O(1)$

The time complexity of our method according to above is $O(nm + n \log n|T| + w^2 + w^3 + \sum_{l=1}^{q} n_{(l)}^3 + n(\sum_{l=1}^{q} n_{(l)}^2(w^2 + n) + \sum_{l=1}^{q} n_{(l)}^2(1 + n^2w^2 \sum_{l=1}^{q} n_{(l)}^2)))$ and the space complexity is $O(|m_1 \cup m_2| + \sum_{l=1}^{q} n_{(l)}^2 + w^2 + nw + (n-w)^2)$.

* $m_1 = |V_1|$ of G_1, $m_2 = |V_2|$ of G_2; T is the number of iteration on slashburn; sparse matrices multiplication ($\mathbf{A}' \times \mathbf{D}'$) is $O(mn)$; $n_{(l)}(\ll n)$ is $|V|$ of l^{th} block diagonal matrix; inversion of $[n \times n]$ matrix: $O(n^3)$; inversion of q-block diagonal matrix: $O(\sum_{l=1}^{q} n_{(l)}^3)$.

applying the proposed method to processing real-world data and compares it with existing state-of-the-art methods. Refer to the caption for Fig. 4 for more details.

We showcase a few implementations of our method for real-world datasets in Fig. 5 to suggest the possibility of applications to various fields. We collected the usage data of public transport SmartCard for a large metropolitan area between July 10 and 16, 2013. For tens of millions of transit passengers each day, the dataset contains passenger identifiers, departure points, departure times, arrival points, arrival times, and accompanying passengers. For analysis, we approached the dataset with both dynamic and static graphing. The flow of passengers through subway stations in this area and its vicinity were put in slots of 10-minute periods. One static graph was created for each time slot covering the seven-day period, and we identified patterns of passenger flow for the same time slots in different days of week. We also traced the dynamic change in the graphs over the course of day to find out how passenger flow differs each day in a week. More details can be found in the caption for Fig. 5.

Table IV lists the complexity of the proposed approach. The analysis assumes that the input graph possesses the characteristics of real-world datasets as in scale-free networks.

We believe that the proposed method provides a significant improvement in terms of time and space complexity over existing state-of-the-art methods, especially when the size of the graph is large. To accommodate very large graphs with hundreds of millions of nodes, our future work includes the consideration of approximation and parallelization [14] along with graph sampling methods.

V. Conclusion

The purpose of this work is to offer an accurate and efficient similarity measure for dynamically changing large-scale graphs and structurally comparable graphs. The proposed method relies on the idea of intergraph compression implemented as the RWR algorithm by the Schur complement and shows significant improvements in terms of time and space complexity for measuring graph similarity over existing state-of-the-art methods. Using the proposed method, medium-sized dynamic graphs can be scrutinized for anomaly in real time, and structural similarity for graphs with millions of nodes can be measured in a relatively short period of time for applications such as grouping nodes for in-depth analysis. Big data has been an emerging key word in recent years, and the proposed method offers an adequate basis for efficiently examining large-scale real-world data in graph forms. Our analysis of real-world usage data of the public transportation SmartCard in a large metropolitan area showcases a successful application of the method, which can further be employed in analyzing data from various fields.

Acknowledgment

This work was supported by the NRF grant funded by the Korea government (Ministry of Science, ICT and Future Planning, MSIP) [No. 2011-0009963], by the ICT R&D program of MSIP/ITP [14-824-09-014, Basic Software Research in Human-level Lifelong Machine Learning], by the Brain Korea 21 Plus Project in 2015, and by SAP Labs Korea.

References

[1] D. J. Cook and L. B. Holder, *Mining graph data*. John Wiley & Sons, 2006.

[2] S. Yoon, J. C. Ebert, E.-Y. Chung, G. De Micheli, and R. B. Altman, “Clustering protein environments for function prediction: finding prosite motifs in 3d,” *BMC bioinformatics*, vol. 8, no. Suppl 4, p. S10, 2007.

[3] D. Koutra, J. T. Vogelstein, and C. Faloutsos, “Deltacon: A principled massive-graph similarity function,” *arXiv:1304.4657*, 2013.

[4] H. Tong, C. Faloutsos, and J.-Y. Pan, “Fast random walk with restart and its applications,” 2006.

[5] V. Bhaskaran and K. Konstantinides, *Image and video compression standards: algorithms and architectures*. Springer Science & Business Media, 1997.

[6] G. H. Golub and C. F. Van Loan, *Matrix computations*, vol. 3. JHU Press, 2012.

[7] K. Shin, J. Jung, S. Lee, and U. Kang, “Bear: Block elimination approach for random walk with restart on large graphs,” in *Proceedings of ACM SIGMOD*, pp. 1571–1585, 2015.

[8] S. Soundarajan, T. Eliassi-Rad, and B. Gallagher, “A guide to selecting a network similarity method,” SIAM, 2014.

[9] J. Leskovec and A. Krevl, “SNAP Datasets: Stanford large network dataset collection.” http://snap.stanford.edu/data, June 2014.

[10] A. N. Langville and C. D. Meyer, *Google’s PageRank and beyond: the science of search engine rankings*. Princeton University Press, 2011.

[11] Y. Fujiwara, M. Nakatsuji, T. Yamamuro, H. Shiokawa, and M. Onizuka, “Efficient personalized pagerank with accuracy assurance,” in *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 15–23, ACM, 2012.

[12] D. Koutra, T.-Y. Ke, U. Kang, D. H. P. Chau, H.-K. K. Pao, and C. Faloutsos, “Unifying guilt-by-association approaches: Theorems and fast algorithms,” in *Machine Learning and Knowledge Discovery in Databases*, pp. 245–260, Springer, 2011.

[13] Y. Lim, U. Kang, and C. Faloutsos, “Slashburn: Graph compression and mining beyond caveman communities,” 2013.

[14] Y. Jeon and S. Yoon, “Multi-threaded hierarchical clustering by parallel nearest-neighbor chaining,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 9, pp. 2534–2548, 2015.