

DELTA CON: A Principled Massive-Graph Similarity Function with Attribution

Danai Koutra, Computer Science Department, Carnegie Mellon University.

Neil Shah, Computer Science Department, Carnegie Mellon University.

Joshua T. Vogelstein, Department of Statistical Science, Duke University.

Brian Gallagher, Lawrence Livermore National Laboratory.

Christos Faloutsos, Computer Science Department, Carnegie Mellon University.

How much did a network change since yesterday? How different is the wiring between Bob's brain (a left-handed male) and Alice's brain (a right-handed female), and how is it different? Graph similarity with given node correspondence, i.e. the detection of changes in the connectivity of graphs, arises in numerous settings. In this work, we formally state the axioms and desired properties of the graph similarity functions, and evaluate when state-of-the-art methods fail to detect crucial connectivity changes in graphs. We propose DELTA CON, a principled, intuitive, and scalable algorithm that assesses the similarity between two graphs on the same nodes (e.g. employees of a company, customers of a mobile carrier). In conjunction, we propose DELTA CON-ATTR, a related approach which enables attribution of change or dissimilarity to responsible nodes and edges. Experiments on various synthetic and real graphs showcase the advantages of our method over existing similarity measures. Finally, we employ DELTA CON and DELTA CON-ATTR on real applications: (a) we classify people to groups of high and low creativity based on their brain connectivity graphs, (b) do temporal anomaly detection in the who-emails-whom Enron graph and find the top culprits for the changes in the temporal corporate email graph, and (c) recover pairs of test-retest large brain scans (~17M edges, up to 90M edges) for 21 subjects.

Categories and Subject Descriptors: H.2.8 [Database Management]: Data Mining

General Terms: Algorithms, Experimentation

Additional Key Words and Phrases: graph similarity, graph comparison, anomaly detection, network monitoring, graph classification, node attribution, edge attribution

ACM Reference Format:

Danai Koutra, Neil Shah, Joshua T. Vogelstein, Brian Gallagher and Christos Faloutsos. DELTA CON: A Principled Massive-Graph Similarity Function with Attributions. *ACM Trans. Knowl. Discov. Data.* V, N, Article XY (May 2014), 35 pages.

DOI: <http://dx.doi.org/10.1145/0000000.0000000>

1. INTRODUCTION

Graphs arise naturally in numerous situations; social, traffic, collaboration and computer networks, images, protein-protein interaction networks, brain connectivity graphs and web graphs are only a few examples. A problem that comes up often in all those settings is the following: how much do two graphs or networks differ in terms of connectivity, and which are the main node and edge culprits for the difference?

Author's address: Danai Koutra, Computer Science Department, School of Computer Science, Carnegie Mellon University, email: danai@cs.cmu.edu.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies show this notice on the first page or initial screen of a display along with the full citation. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, to redistribute to lists, or to use any component of this work in other works requires prior specific permission and/or a fee. Permissions may be requested from Publications Dept., ACM, Inc., 2 Penn Plaza, Suite 701, New York, NY 10121-0701 USA, fax +1 (212) 869-0481, or permissions@acm.org.

© 2014 ACM 1556-4681/2014/05-ARTXY \$15.00

DOI: <http://dx.doi.org/10.1145/0000000.0000000>

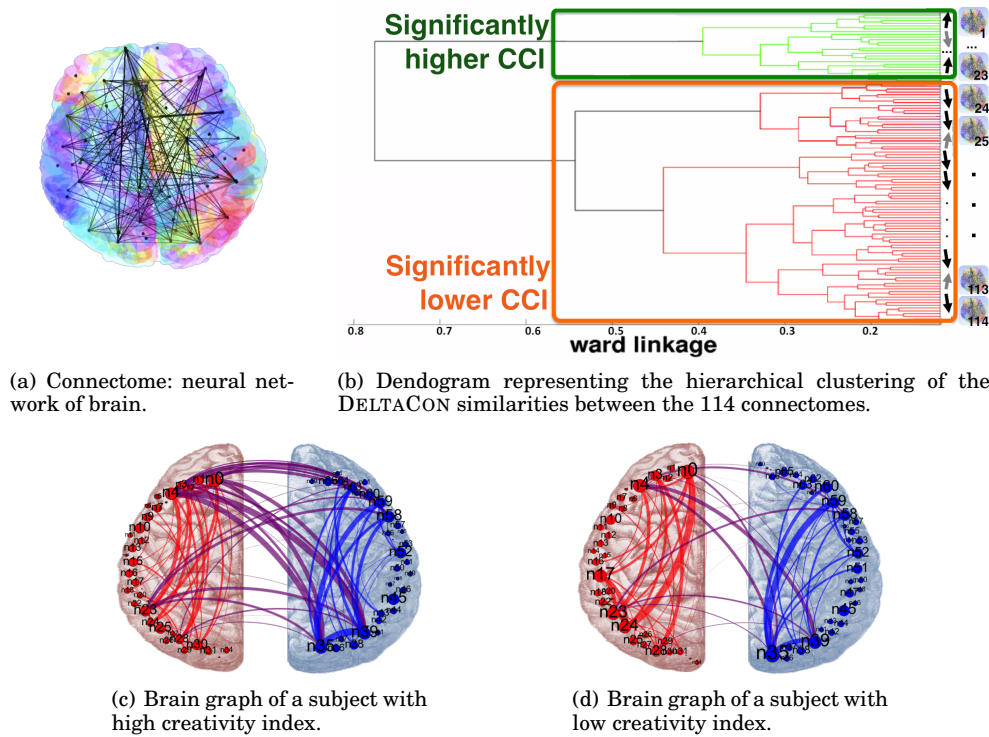


Fig. 1: DELTACON-based clustering and classification shows that artistic brains seem to have different wiring than the rest. (a) Brain network (connectome). Different colors correspond to each of the 70 cortical regions, whose centers are depicted by vertices. Connections between the regions are shown by edges. (b) The connectomes are classified in two big clusters by hierarchical clustering. The classification is based on the pairwise DELTACON similarities between the 114 connectomes that we study. Elements in red correspond to mostly high creativity score. (c)-(d) Illustration of brain graphs for subjects with low and high creativity index respectively. The low-CCI brain has fewer and lighter cross-hemisphere connections than the high-CCI brain.

Similarity or comparison of aligned graphs (i.e., with known node correspondence) is a core task for sense-making: abnormal changes in the network traffic may indicate a computer attack; differences of big extent in a who-calls-whom graph may reveal a national celebration, or a telecommunication problem. Besides, network similarity can serve as a building block for similarity-based classification [Chen et al. 2009] of graphs, and give insights into transfer learning, as well as behavioral patterns: is the Facebook message graph similar to the Facebook wall-to-wall graph? Tracking changes in networks over time, spotting anomalies and detecting events is a research direction that has attracted much interest (e.g., [Caceres et al. 2011], [Noble and Cook 2003], [Wang et al. 2011]).

Long in the purview of researchers, graph similarity is a well-studied problem and several approaches have been proposed to solve variations of the problem. However, graph comparison with node/edge attribution still remains an open problem, while, with the passage of time, the list of requirements increases: the exponential growth of graphs, both in number and size, calls for methods that are not only accurate, but also scalable to graphs with billions of nodes.

In this paper, we address three main problems: How to compare two networks efficiently, how to evaluate the degree of their similarity and how to identify the culprit nodes/edges responsible for differences. Our main contributions are the following:

- (1) *Axioms/Properties*: We formalize the axioms and properties that a similarity measure must conform to.
- (2) *Algorithm*: We propose DELTACON for measuring connectivity differences between two graphs, and show that it is: (a) *principled*, conforming to all the axioms presented in Section 2, (b) *intuitive*, giving similarity scores that agree with common sense and can be easily explained, and (c) *scalable*, able to handle large-scale graphs. We also introduce DELTACON-ATTR for change attribution between graphs.
- (3) *Experiments*: we report experiments on synthetic and real datasets, and compare our similarity measure to six state-of-the-art methods that apply to our setting.
- (4) *Applications*: We use DELTACON for real-world applications, such as temporal anomaly detection and clustering/classification. In Fig. 1, DELTACON is used for clustering brain graphs corresponding to 114 individuals; the two big clusters which differ in terms of connectivity correspond to people with significantly different level of creativity. More details are given in Sec. 6.

This paper is an extension of the conference paper [Koutra et al. 2013b]. There are several differences from its earlier version: (i) we introduce the problem of attributing the detected difference between two graphs to nodes and edges, and give two algorithms, DELTACON-ATTR for Node Attribution and DELTACON-ATTR for Edge Attribution, that build on top of our originally proposed graph similarity method in order to find the culprit nodes and edges (Section 4); (ii) we give the proofs of our lemmas and theorems, and discuss the mathematical details for the satisfiability of the axioms and desired properties by DELTACON (Sections 3.1-3.3); (iii) we include experiments for our attribution algorithm, DELTACON-ATTR, on small synthetic datasets, as well as on real, temporal graphs, and report interesting findings (Section 5.2); (iv) we report more experiments for our graph similarity method, DELTACON, on additional large-scale real networks, and include one more application on large brain graphs spanning up to 90M edges (Sections 5-6); and (v) we provide a more extensive survey of the related work (Section 7).

The paper is organized as follows: Section 2 presents the intuition behind our main method, and the axioms and desired properties of a similarity measure; Section 3 and 4 have the proposed algorithms for similarity computation and node/edge attribution, as well as theoretical proofs for the axioms and properties; experiments on synthetic and big real networks are in Section 5; Section 6 presents three real-world applications; the related work and the conclusions are in Section 7 and 8 respectively. Finally, Table I presents the major symbols we use in the paper and their definitions.

2. DELTACON: INTUITION

How can we find the similarity in connectivity between two graphs, or more formally how can we solve the following problem?

PROBLEM DEFINITION. [DELTACONnectivity]

Given. (a) two graphs, $G_1(\mathcal{V}, \mathcal{E}_1)$ and $G_2(\mathcal{V}, \mathcal{E}_2)$ with the same node set¹, \mathcal{V} , but different edge sets \mathcal{E}_1 and \mathcal{E}_2 , and (b) the node correspondence.

Find. a similarity score, $sim(G_1, G_2) \in [0, 1]$, between the input graphs. Similarity score of value 0 means totally different graphs, while 1 means identical graphs.

¹If the graphs have different, but overlapping, node sets \mathcal{V}_1 and \mathcal{V}_2 , we assume that $\mathcal{V} = \mathcal{V}_1 \cup \mathcal{V}_2$, and the extra nodes are treated as singletons.

Table I: Symbols and Definitions. Bold capital letters: matrices, lowercase letters with arrows: vectors, plain font: scalars.

Symbol	Description
G	graph
\mathcal{V}, n	set of nodes, number of nodes
\mathcal{E}, m	set of edges, number of edges
$sim(G_1, G_2)$	similarity between graphs G_1 and G_2
$d(G_1, G_2)$	distance between graphs G_1 and G_2
I	$n \times n$ identity matrix
A	$n \times n$ adjacency matrix with elements a_{ij}
D	$n \times n$ diagonal degree matrix, $d_{ii} = \sum_j a_{ij}$
L	$= \mathbf{D} - \mathbf{A}$ laplacian matrix
S	$n \times n$ matrix of final scores with elements s_{ij}
S'	$n \times g$ reduced matrix of final scores
\vec{e}_i	$n \times 1$ unit vector with 1 in the i^{th} element
\vec{s}_{0k}	$n \times 1$ vector of seed scores for group k
\vec{s}_i	$n \times 1$ vector of final affinity scores to node i
g	number of groups (node partitions)
ϵ	$= 1/(1 + \max_i (d_{ii}))$ positive constant (< 1) encoding the influence between neighbors
DC₀, DC	DELTA _{CON} ₀ , DELTA _{CON}
VEO	Vertex/Edge Overlap
GED	Graph Edit Distance [Bunke et al. 2006]
SS	Signature Similarity [Papadimitriou et al. 2008]
λ-D Adj.	λ -distance on the Adjacency A
λ-D LAP	λ -distance on the Laplacian L
λ-D N.L.	λ -distance on the normalized Laplacian L

The obvious way to solve this problem is by measuring the overlap of their edges. Why does this often not work in practice? Consider the following example. According to the overlap method, the pairs of barbell graphs shown in Fig. 3 of p. 17, $(B10, mB10)$ and $(B10, mmB10)$, have the same similarity score. But, clearly, from the aspect of information flow, a missing edge from a clique ($mB10$) does not play as important role in the graph connectivity as the missing “bridge” in $mmB10$. So, could we instead measure the differences in the 1-step away neighborhoods, 2-step away neighborhoods etc.? If yes, with what weight? It turns out (Intuition 1, p. 6) that our method does exactly this in a principled way.

2.1. Fundamental Concept

The first conceptual step of our proposed method is to compute the pairwise node affinities in the first graph, and compare them with the ones in the second graph. For notational compactness, we store them in a $n \times n$ similarity matrix² **S**. The s_{ij} entry of the matrix indicates the influence node i has on node j . For example, in a who-knows-whom network, if node i is, say, republican and if we assume homophily (i.e., neighbors are similar), how likely is it that node j is also republican? Intuitively, node i has more influence/affinity to node j if there are many, short, heavily weighted paths from node i to j .

The second conceptual step is to measure the differences in the corresponding node affinity scores of the two graphs and report the result as their similarity score.

2.2. How to measure node affinity?

Pagerank [Brin and Page 1998], personalized Random Walks with Restarts (RWR) [Haveliwala 2003], lazy RWR [Aldous and Fill 2002], and the “electrical network analogy” technique [Doyle and Snell 1984] are only a few of the methods that compute node affinities. We could have used

²In reality, we don’t measure all the affinities (see Section 3.2 for an efficient approximation).

Personalized RWR: $[\mathbf{I} - (1 - c)\mathbf{A}\mathbf{D}^{-1}]\vec{s}_i = c\vec{e}_i$, where c is the probability of restarting the random walk from the initial node, \vec{e}_i the starting (seed) indicator vector (all zeros except 1 at position i), and \vec{s}_i the unknown Personalized Pagerank column vector. Specifically, s_{ij} is the affinity of node j w.r.t. node i . For reasons that we explain next, we chose to use a more recent and principled method, the so-called Fast Belief Propagation (FABP) [Koutra et al. 2011], and specifically a simplified form of it given by:

$$[\mathbf{I} + \epsilon^2\mathbf{D} - \epsilon\mathbf{A}]\vec{s}_i = \vec{e}_i \quad (1)$$

where $\vec{s}_i = [s_{i1}, \dots, s_{in}]^T$ is the column vector of final similarity/influence scores starting from the i^{th} node, ϵ is a small constant capturing the influence between neighboring nodes, \mathbf{I} is the identity matrix, \mathbf{A} is the adjacency matrix and \mathbf{D} is the diagonal matrix with the degree of node i as the d_{ii} entry.

An equivalent, more compact notation, is to use a matrix form, and to stack all the \vec{s}_i vectors ($i = 1, \dots, n$) into the $n \times n$ matrix \mathbf{S} . We can easily prove that

$$\mathbf{S} = [\mathbf{S}_{ij}] = [\mathbf{I} + \epsilon^2\mathbf{D} - \epsilon\mathbf{A}]^{-1}. \quad (2)$$

Derivations and Equivalences. Before we give the reasons we use Belief Propagation in order to measure node affinity, we provide the next observation which briefly explains the simplification in the FABP equation that we used.

OBSERVATION 1.(From FABP to DELTACON- Sketch of Derivation) We start from the FABP [Koutra et al. 2011], which is a fast approximation of the loopy BP algorithm, and is guaranteed to converge: $[\mathbf{I} + a\mathbf{D} - c'\mathbf{A}]\vec{s} = \vec{s}_0$, where \vec{s}_0 is the vector of prior scores, \vec{s} is the vector of final scores (beliefs), $a = 4h_h^2/(1 - 4h_h^2)$, and $c' = 2h_h/(1 - 4h_h^2)$ are small constants, and h_h is a small constant that encodes the influence between neighboring nodes. By (a) using the MacLaurin expansion for $1/(1 - 4h_h^2) \approx 1 + 4h_h^2$ and omitting the terms of power greater than 2, (b) setting $\vec{s}_0 = \vec{e}_i$ and $\vec{s} = \vec{s}_i$, and (c) setting $h_h = \epsilon/2$, we obtain eq. (1), the core formula of DELTACON.

Finally, for the reader who is familiar with RWR, we note that the version of FABP that we use is identical to Personalized RWR under specific conditions, as shown in Theorem 1.

THEOREM 1.

The FABP equation (1) can be written in the Personalized RWR-like form:

$$[\mathbf{I} - (1 - c'')\mathbf{A}_*\mathbf{D}^{-1}]\vec{s}_i = c''\vec{y},$$

where $c'' = 1 - \epsilon$, $\vec{y} = \mathbf{A}_*\mathbf{D}^{-1}\mathbf{A}^{-1}\frac{1}{c''}\vec{e}_i$ and $\mathbf{A}_* = \mathbf{D}(\mathbf{I} + \epsilon^2\mathbf{D})^{-1}\mathbf{D}^{-1}\mathbf{A}\mathbf{D}$.

PROOF. We begin from the derived FABP equation (1) and do simple linear algebra operations:

$$\begin{aligned} [\mathbf{I} + \epsilon^2\mathbf{D} - \epsilon\mathbf{A}]\vec{s}_i &= \vec{e}_i & (\times \mathbf{D}^{-1} \text{ from the left}) \\ [\mathbf{D}^{-1} + \epsilon^2\mathbf{I} - \epsilon\mathbf{D}^{-1}\mathbf{A}]\vec{s}_i &= \mathbf{D}^{-1}\vec{e}_i & (\mathbf{F} = \mathbf{D}^{-1} + \epsilon^2\mathbf{I}) \\ [\mathbf{F} - \epsilon\mathbf{D}^{-1}\mathbf{A}]\vec{s}_i &= \mathbf{D}^{-1}\vec{e}_i & (\times \mathbf{F}^{-1} \text{ from the left}) \\ [\mathbf{I} - \epsilon\mathbf{F}^{-1}\mathbf{D}^{-1}\mathbf{A}]\vec{s}_i &= \mathbf{F}^{-1}\mathbf{D}^{-1}\vec{e}_i & (\mathbf{A}_* = \mathbf{F}^{-1}\mathbf{D}^{-1}\mathbf{A}\mathbf{D}) \\ [\mathbf{I} - \epsilon\mathbf{A}_*\mathbf{D}^{-1}]\vec{s}_i &= (1 - \epsilon)(\mathbf{A}_*\mathbf{D}^{-1}\mathbf{A}^{-1}\frac{1}{1-\epsilon}\vec{e}_i) \quad \square \end{aligned}$$

2.3. Why use Belief Propagation?

The reasons we choose BP and its fast approximation with Eq. (2) are: (a) it is based on sound theoretical background (maximum likelihood estimation on marginals), (b) it is fast (linear on the number of edges), and (c) it agrees with intuition, taking into account not only direct neighbors, but also 2-, 3- and k -step-away neighbors, with decreasing weight. We elaborate on the last reason, next:

INTUITION 1.(Attenuating Neighboring Influence)

By temporarily ignoring the term $\epsilon^2 \mathbf{D}$ in (2), we can expand the matrix inversion and approximate the $n \times n$ matrix of pairwise affinities, \mathbf{S} , as

$$\mathbf{S} \approx [\mathbf{I} - \epsilon \mathbf{A}]^{-1} \approx \mathbf{I} + \epsilon \mathbf{A} + \epsilon^2 \mathbf{A}^2 + \dots$$

As we said, our method captures the differences in the 1-step, 2-step, 3-step etc. neighborhoods in a weighted way; differences in long paths have smaller effect on the computation of the similarity measure than differences in short paths. Recall that $\epsilon < 1$, and that \mathbf{A}^k has information about the k -step paths. Notice that this is just the intuition behind our method; we do not use this simplified formula to find matrix \mathbf{S} .

2.4. Which properties should a similarity measure satisfy?

Let $G_1(\mathcal{V}, \mathcal{E}_1)$ and $G_2(\mathcal{V}, \mathcal{E}_2)$ be two graphs, and $\text{sim}(G_1, G_2) \in [0, 1]$ denote their similarity score. Then, we want the similarity measure to obey the following axioms:

- A1. *Identity property*: $\text{sim}(G_1, G_1) = 1$
- A2. *Symmetric property*: $\text{sim}(G_1, G_2) = \text{sim}(G_2, G_1)$
- A3. *Zero property*: $\text{sim}(G_1, G_2) \rightarrow 0$ for $n \rightarrow \infty$, where G_1 is the complete graph (K_n), and G_2 is the empty graph (i.e., the edge sets are complementary).

Moreover, the measure must be:

(a) intuitive. It should satisfy the following desired properties:

P1. [*Edge Importance*] For unweighted graphs, changes that create disconnected components should be penalized more than changes that maintain the connectivity properties of the graphs.

P2. [*Edge-“Submodularity”*] For unweighted graphs, a specific change is more important in a graph with few edges than in a much denser, but equally sized graph.

P3. [*Weight Awareness*] In weighted graphs, the bigger the weight of the removed edge is, the greater the impact on the similarity measure should be.

In Section 3.3, we formalize the properties and discuss their satisfiability by our proposed similarity measure theoretically. Moreover, in Section 5 we introduce and discuss about an additional, *informal*, property:

IP. [*Focus Awareness*] “Random” changes in graphs are less important than “targeted” changes of the same extent.

(b) scalable. The huge size of the generated graphs, as well as their abundance require a similarity measure that is computed fast and handles graphs with billions of nodes.

3. DELTACON: DETAILS

Now that we have described the high level ideas behind our method, we move on to the details.

3.1. Algorithm Description

Let the graphs we compare be $G_1(\mathcal{V}, \mathcal{E}_1)$ and $G_2(\mathcal{V}, \mathcal{E}_2)$. If the graphs have different node sets, say \mathcal{V}_1 and \mathcal{V}_2 , we assume that $\mathcal{V} = \mathcal{V}_1 \cup \mathcal{V}_2$, where some nodes are disconnected.

As mentioned before, the main idea behind our proposed similarity algorithm is to compare the node affinities in the given graphs. The steps of our similarity method are:

Step 1. By eq. (2), we compute for each graph the $n \times n$ matrix of pairwise node affinity scores (\mathbf{S}_1 and \mathbf{S}_2 for graphs G_1 and G_2 respectively).

Step 2. Among the various distance and similarity measures (e.g., Euclidean distance (ED), cosine similarity, correlation) found in the literature, we use the root Euclidean distance

(ROOTED, a.k.a. Matusita distance)

$$d = \text{ROOTED}(\mathbf{S}_1, \mathbf{S}_2) = \sqrt{\sum_{i=1}^n \sum_{j=1}^n (\sqrt{s_{1,ij}} - \sqrt{s_{2,ij}})^2}. \quad (3)$$

We use the ROOTED distance for the following reasons:

- (1) it is very similar to the Euclidean distance (ED), the only difference being the square root of the pairwise similarities (s_{ij}),
- (2) it usually gives better results, because it “boosts” the node affinities³ and, therefore, detects even small changes in the graphs (other distance measures, including ED, suffer from high similarity scores no matter how much the graphs differ), and
- (3) satisfies the desired properties $P1$ - $P3$, as well as the informal property IP . As discussed in Section 3.3, at least $P1$ is not satisfied by the ED.

Step 3. For interpretability, we convert the distance (d) to a similarity measure (sim) via the formula $sim = \frac{1}{1+d}$. The result is bounded to the interval $[0,1]$, as opposed to being unbounded $[0,\infty)$. Notice that the distance-to-similarity transformation does *not* change the ranking of results in a nearest-neighbor query.

The straightforward algorithm, DELTACON₀ (Algorithm 1), is to compute all the n^2 affinity scores of matrix \mathbf{S} by simply using equation (2). We can do the inversion using the Power Method or any other efficient method.

ALGORITHM 1: DELTACON₀

INPUT: edge files of $G_1(\mathcal{V}, \mathcal{E}_1)$ and $G_2(\mathcal{V}, \mathcal{E}_2)$
 // $\mathcal{V} = \mathcal{V}_1 \cup \mathcal{V}_2$, if \mathcal{V}_1 and \mathcal{V}_2 are the graphs' node sets
 $\mathbf{S}_1 = [\mathbf{I} + \epsilon^2 \mathbf{D}_1 - \epsilon \mathbf{A}_1]^{-1}$ // $s_{1,ij}$: affinity/influence of
 $\mathbf{S}_2 = [\mathbf{I} + \epsilon^2 \mathbf{D}_2 - \epsilon \mathbf{A}_2]^{-1}$ // node i to node j in G_1
 $d(G_1, G_2) = \text{ROOTED}(\mathbf{S}_1, \mathbf{S}_2)$
return $sim(G_1, G_2) = \frac{1}{1+d(G_1, G_2)}$

3.2. Speeding up: DELTACON

DELTACON₀ satisfies all the properties in Section 2, but it is quadratic (n^2 affinity scores s_{ij} are computed by using the power method for the inversion of a sparse matrix) and thus not scalable. We present a faster, linear algorithm, DELTACON (Algorithm 2), which approximates DELTACON₀ and differs in the first step. We still want each node to become a seed exactly once in order to find the affinities of the rest of the nodes to it; but, here, we have multiple seeds at once, instead of having one seed at a time. The idea is to randomly divide our node-set into g groups, and compute the affinity score of each node i to group k , thus requiring only $n \times g$ scores, which are stored in the $n \times g$ matrix \mathbf{S}' ($g \ll n$). Intuitively, instead of using the $n \times n$ affinity matrix \mathbf{S} , we add up the scores of the columns that correspond to the nodes of a group, and obtain the $n \times g$ matrix \mathbf{S}' ($g \ll n$). The score s'_{ik} is the affinity of node i to the k^{th} group of nodes ($k = 1, \dots, g$). The following lemma gives the complexity of computing the node-group affinities.

LEMMA 1.

The time complexity of computing the reduced affinity matrix, \mathbf{S}' , is linear on the number of edges.

³The node affinities are in $[0, 1]$, so the square root makes them bigger.

PROOF. We can compute the $n \times g$ “skinny” matrix S' quickly, by solving $[I + \epsilon^2 D - \epsilon A]S' = [\vec{s}_{01} \dots \vec{s}_{0g}]$, where $\vec{s}_{0k} = \sum_{i \in \text{group}_k} \vec{e}_i$ is the membership $n \times 1$ vector for group k (all 0's, except 1's for members of the group). Solving this system is equivalent to solving for each group g the linear system $[I + \epsilon^2 D - \epsilon A]S' = \vec{s}_{0g}$. The latter can be solved –using the Power Method [Koutra et al. 2011]– in time linear on the number of non-zeros of the matrix $\epsilon A - \epsilon^2 D$, which is equivalent to the number of edges, m , of the input graph G . Thus, the g linear systems require $O(g \cdot m)$ time, which is still linear on the number of edges for a small constant g . It is worth noting that the g linear systems can be solved in parallel, since there are no dependencies, and then the overall time is simply $O(m)$. \square

Thus, we compute g final scores per node, which denote its affinity to every *group* of seeds, instead of every seed node that we had in eq. (2). With careful implementation, DELTACON is linear on the number of edges and groups g . As we show in section 5.3, it takes $\sim 160\text{sec}$, on commodity hardware, for a 1.6-million-node graph. Once we have the reduced affinity matrices S'_1 and S'_2 of the two graphs, we use the ROOTED, to find the similarity between the $n \times g$ matrices of final scores, where $g \ll n$. The pseudocode of the DELTACON is given in Algorithm 2.

ALGORITHM 2: DELTACON

INPUT: edge files of $G_1(\mathcal{V}, \mathcal{E}_1)$ and $G_2(\mathcal{V}, \mathcal{E}_2)$ and
 g (groups: # of node partitions)

$\{\mathcal{V}_j\}_{j=1}^g = \text{random_partition}(\mathcal{V}, g)$ // g groups
 // estimate affinity vector of nodes $i = 1, \dots, n$ to group k
for $k = 1 \rightarrow g$ **do**
 $\vec{s}_{0k} = \sum_{i \in \mathcal{V}_k} \vec{e}_i$
 solve $[I + \epsilon^2 D_1 - \epsilon A_1] \vec{s}'_{1k} = \vec{s}_{0k}$
 solve $[I + \epsilon^2 D_2 - \epsilon A_2] \vec{s}'_{2k} = \vec{s}_{0k}$
end for
 $S'_1 = [\vec{s}'_{11} \ \vec{s}'_{12} \ \dots \ \vec{s}'_{1g}]$; $S'_2 = [\vec{s}'_{21} \ \vec{s}'_{22} \ \dots \ \vec{s}'_{2g}]$
 // compare affinity matrices S'_1 and S'_2
 $d(G_1, G_2) = \text{ROOTED}(S'_1, S'_2)$
return $\text{sim}(G_1, G_2) = \frac{1}{1+d(G_1, G_2)}$

In an attempt to see how our random node partitioning algorithm in the first step fares with respect to more principled partitioning techniques, we used METIS [Karypis and Kumar 1995]. Essentially, such an approach finds the influence of *coherent* subgraphs to the rest of the nodes in the graph – instead of the influence of randomly chosen nodes to the latter. We found that the METIS-based variant of our similarity method gave intuitive results for most small, synthetic graphs, but not for the real graphs. This is probably related to the lack of good edge-cuts on highly sparse real graphs, and also the fact that changes within a community manifest less when the influence of that community to the rest of the nodes is measured than when the participating nodes are randomly assigned to different groups.

Next we give the time complexity of DELTACON, as well as the relationship between the similarity scores of DELTACON₀ and DELTACON.

LEMMA 2.

The time complexity of DELTACON, when applied in parallel to the input graphs, is linear on the number of edges in the graphs, i.e. $O(g \cdot \max\{m_1, m_2\})$.

PROOF. By using the Power Method [Koutra et al. 2011], the complexity of solving eq. (1) is $O(m_i)$ for each graph ($i = 1, 2$). The node partitioning needs $O(n)$ time; the affinity algorithm is run g times in each graph, and the similarity score is computed in $O(gn)$ time. Therefore, the complexity of DELTACON is $O((g+1)n + g(m_1 + m_2))$, where g is a small constant. Unless the graphs are trees, $|\mathcal{E}_i| < n$, so the complexity of the algorithm reduces to $O(g(m_1 + m_2))$. Assuming

that the affinity algorithm is run on the graphs in parallel, since there is no dependency between the computations, DELTACON has complexity $O(g \cdot \max\{m_1, m_2\})$. \square

Before we give the relationship between the similarity scores computed by the two proposed methods, we introduce a helpful lemma.

LEMMA 3.

The affinity score of each node to a group (computed by DELTACON) is equal to the sum of the affinity scores of the node to each one of the nodes in the group individually (computed by DELTACON₀).

PROOF. Let $\mathbf{B} = \mathbf{I} + \epsilon^2 \mathbf{D} - \epsilon \mathbf{A}$. Then DELTACON₀ consists of solving for every node $i \in \mathcal{V}$ the equation $\mathbf{B}\vec{s}_i = \vec{e}_i$; DELTACON solves the equation $\mathbf{B}\vec{s}'_k = \vec{s}_{0k}$ for all groups $k \in (0, g]$, where $\vec{s}_{0k} = \sum_{i \in \text{group}_k} \vec{e}_i$. Because of the linearity of matrix additions, it holds true that $\vec{s}'_k = \sum_{i \in \text{group}_k} \vec{s}_i$, for all groups k . \square

THEOREM 2.

DELTACON's similarity score between any two graphs G_1, G_2 upper bounds the actual DELTACON₀'s similarity score, i.e. $\text{sim}_{DC_0}(G_1, G_2) \leq \text{sim}_{DC}(G_1, G_2)$.

PROOF. Intuitively, grouping nodes blurs the influence information and makes the nodes seem more similar than originally.

More formally, let $\mathbf{S}_1, \mathbf{S}_2$ be the $n \times n$ final-scores matrices of G_1 and G_2 by applying DELTACON₀, and $\mathbf{S}'_1, \mathbf{S}'_2$ be the respective $n \times g$ final-scores matrices by applying DELTACON. We want to show that DELTACON₀'s distance

$$d_{DC_0} = \sqrt{\sum_{i=1}^n \sum_{j=1}^n (\sqrt{s_{1,ij}} - \sqrt{s_{2,ij}})^2}$$

is greater than DELTACON's distance

$$d_{DC} = \sqrt{\sum_{k=1}^g \sum_{i=1}^n (\sqrt{s'_{1,ik}} - \sqrt{s'_{2,ik}})^2}$$

or, equivalently, that $d_{DC_0}^2 > d_{DC}^2$. It is sufficient to show that for one group of DELTACON, the corresponding summands in d_{DC} are smaller than the summands in d_{DC_0} which are related to the nodes that belong to the group. By extracting the terms in the squared distances that refer to one group of DELTACON and its member nodes in DELTACON₀, and by applying Lemma 3, we obtain the following terms:

$$\begin{aligned} t_{DC_0} &= \sum_{i=1}^n \sum_{j \in \text{group}} (\sqrt{s_{1,ij}} - \sqrt{s_{2,ij}})^2 \\ t_{DC} &= \sum_{i=1}^n (\sqrt{\sum_{j \in \text{group}} s_{1,ij}} - \sqrt{\sum_{j \in \text{group}} s_{2,ij}})^2. \end{aligned}$$

Next we concentrate again on a selection of summands (e.g. $i = 1$), we expand the squares and use the Cauchy-Schwartz inequality to show that

$$\sum_{j \in \text{group}} \sqrt{s_{1,ij} s_{2,ij}} < \sqrt{\sum_{j \in \text{group}} s_{1,ij}} \sqrt{\sum_{j \in \text{group}} s_{2,ij}},$$

or equivalently that $t_{DC_0} > t_{DC}$. \square

3.3. Properties of DELTACON

We have already presented in detail our scalable algorithm for graph similarity, and the only thing that remains from a theoretic perspective is whether DELTACON satisfies the axioms and properties presented in Sec. 2.4.

A1. Identity Property: $\text{sim}(G_1, G_1) = 1$.

The affinity scores, S , are identical for the input graphs, because the two linear systems in Algorithm 1 are exactly the same. Thus, the ROOTED distance d is 0, and the DELTACON similarity, $sim = \frac{1}{1+d}$, is equal to 1.

A2. Symmetric Property: $sim(G_1, G_2) = sim(G_2, G_1)$.

Similarly, the equations that are used to compute $sim(G_1, G_2)$ and $sim(G_2, G_1)$ are the same. The only difference is the order of solving them in Algorithm 1. Therefore, both the ROOTED distance d and the DELTACON similarity score sim are the same.

A3. Zero Property: $sim(G_1, G_2) \rightarrow 0$ for $n \rightarrow \infty$, where G_1 is the complete graph (K_n), and G_2 is the empty graph (i.e., the edge sets are complementary).

PROOF. We restrict ourselves to a sketch of proof, since the proof is rather intricate. First we show that all the nodes in a complete graph get final scores in $\{s_g, s_{ng}\}$, depending on whether they are included in group g or not. Then, it can be demonstrated that the scores have finite limits, and specifically $\{s_g, s_{ng}\} \rightarrow \{\frac{n}{2g} + 1, \frac{n}{2g}\}$ as $n \rightarrow \infty$ (for finite $\frac{n}{g}$). Given this condition, it can be derived that the ROOTED, $d(G_1, G_2)$, between the S matrices of the empty and the complete graph becomes arbitrarily large. So, $sim(G_1, G_2) = \frac{1}{1+d(G_1, G_2)} \rightarrow 0$ for $n \rightarrow \infty$. \square

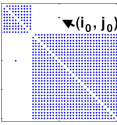
DELTACON satisfies the three axioms that every similarity measure must obey. We elaborate on the satisfiability of the properties of $P1 - P3$ next.

P1. [Edge Importance] For unweighted graphs, changes that create disconnected components should be penalized more than changes that maintain the connectivity properties of the graphs.

Formalizing this property in its most general case with any type of disconnected components is hard, thus we focus on a well-understood and intuitive case: the barbell graph.

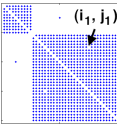
PROOF. Assume that A is the adjacency matrix of an undirected barbell graph with two cliques of size n_1 and n_2 respectively (e.g., B10 with $n_1 = n_2 = 5$ in Fig. 3) and (i_0, j_0) is the “bridge” edge. Without loss of generality we can assume that A has a block-diagonal form, with one edge (i_0, j_0) linking the two blocks:

$$a_{ij} = \begin{cases} 1 & \text{for } i, j \in \{1, \dots, n_1\} \text{ and } i \neq j \\ & \text{or } i, j \in \{n_1 + 1, \dots, n_2\} \text{ and } i \neq j \\ & \text{or } (i, j) = (i_0, j_0) \text{ or } (i, j) = (j_0, i_0) \\ 0 & \text{otherwise} \end{cases}$$

$$A = \begin{bmatrix} \text{clique}_1 & & \\ & \text{clique}_2 & \\ & & \text{bridge} \end{bmatrix}$$


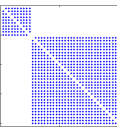
Then, B is an adjacency matrix with elements

$$b_{ij} = \begin{cases} 0 & \text{for one pair } (i, j) \neq (i_0, j_0) \text{ and } (j_0, i_0) \\ a_{ij} & \text{otherwise} \end{cases}$$

$$B = \begin{bmatrix} \text{clique}_1 & & \\ & \text{clique}_2 & \\ & & \text{bridge} \end{bmatrix}$$


and C is an adjacency matrix with elements

$$c_{ij} = \begin{cases} 0 & \text{if } (i, j) = (i_0, j_0) \text{ or } (i, j) = (j_0, i_0) \\ a_{ij} & \text{otherwise} \end{cases}$$

$$C = \begin{bmatrix} \text{clique}_1 & & \\ & \text{clique}_2 & \\ & & \text{bridge} \end{bmatrix}$$


We want to prove that $sim(A, B) \geq sim(A, C) \Leftrightarrow d(A, B) \leq d(A, C)$, or equivalently that $d^2(A, B) \leq d^2(A, C)$.

From eq. (1), by expressing the matrix inversion using a power series and ignoring the terms of greater than second power, for matrix A we obtain the solution:

$$\vec{s}_i = [I + (\epsilon A - \epsilon^2 D_A) + (\epsilon A - \epsilon^2 D_A)^2 + \dots] \vec{e}_i \Rightarrow S_A = I + \epsilon A + \epsilon^2 A^2 - \epsilon^2 D_A,$$

where the index (e.g., A) denotes the graph each matrix corresponds to. Now, using the last equation, we can write out the elements of the S_A, S_B, S_C matrices, and derive their ROOTED

distances:

$$d^2(\mathbf{A}, \mathbf{B}) = 4(n_2 - f) \frac{\epsilon^4}{c_1^2} + 2 \frac{\epsilon^2}{c_2^2}$$

and

$$d^2(\mathbf{A}, \mathbf{C}) = 2(n_1 + n_2 - 2)\epsilon^2 + 2\epsilon,$$

where $c_1 = \sqrt{\epsilon + \epsilon^2(n_2 - 3)} + \sqrt{\epsilon + \epsilon^2(n_2 - 2)}$ and $c_2 = \sqrt{\epsilon^2(n_2 - 2)} + \sqrt{\epsilon + \epsilon^2(n_2 - 2)}$, and $f = 3$ if the missing edge in graph B and the “bridge” edge are incident to the same node, or $f = 2$ in any other case.

We can therefore write the difference of distances we are interested in as

$$d^2(\mathbf{A}, \mathbf{C}) - d^2(\mathbf{A}, \mathbf{B}) = 2\epsilon \left(\epsilon(n_1 + n_2 - f) + 1 - \left(\frac{2\epsilon^3(n_1 - 2)}{c_1^2} + \frac{\epsilon}{c_2^2} \right) \right).$$

By observing that $c_1 \geq 2\sqrt{\epsilon}$ and $c_2 \geq \sqrt{\epsilon}$ for $n_2 \geq 3$ and using these facts in the last equation, it follows that:

$$d^2(\mathbf{A}, \mathbf{C}) - d^2(\mathbf{A}, \mathbf{B}) \geq 2\epsilon \left(\epsilon(n_1 + n_2 - f) - \frac{\epsilon^2(n_1 - 2)}{2} \right) \geq 2\epsilon^2 (n_1 + n_2 - f - \epsilon(n_1 - 2)).$$

Given that $n_2 \geq 3$ and $f = 2$ or 3 , we obtain that $n_2 - f \geq 0$. Moreover, $0 < \epsilon < 1$ by definition and $n_1 - \epsilon n_1 \geq 0$. From these inequalities, it immediately follows that $d^2(\mathbf{A}, \mathbf{B}) \leq d^2(\mathbf{A}, \mathbf{C})$. We note that this property is not always satisfied by the Euclidean distance. \square

P2. [Edge-“Submodularity”] For unweighted graphs, a specific change is more important in a graph with few edges than in a much denser, but equally sized graph.

Let \mathbf{A} be the adjacency matrix of an undirected graph, with $m_{\mathbf{A}}$ non-zero elements a_{ij} and $a_{i_0 j_0} = 1$, and \mathbf{B} be the adjacency matrix of another graph which is identical to \mathbf{A} , but is missing the edge (i_0, j_0) .

$$b_{ij} = \begin{cases} 0 & \text{if } (i, j) = (i_0, j_0) \text{ or } (i, j) = (j_0, i_0) \\ a_{ij} & \text{otherwise} \end{cases}$$

Let’s also assume another pair of graphs \mathbf{C} and \mathbf{H} ⁴ defined as follows:

$$c_{ij} = \begin{cases} 0 & \text{for } \geq 1 \text{ pair } (i, j) \neq (i_0, j_0) \text{ and } (i, j) \neq (j_0, i_0) \\ a_{ij} & \text{otherwise} \end{cases}$$

that is \mathbf{C} has $m_{\mathbf{C}} < m_{\mathbf{A}}$ non-zero elements and

$$h_{ij} = \begin{cases} 0 & \text{for } (i, j) = (i_0, j_0) \text{ or } (i, j) = (j_0, i_0) \\ c_{ij} & \text{otherwise} \end{cases}$$

We want to show that $\text{sim}(\mathbf{A}, \mathbf{B}) \geq \text{sim}(\mathbf{C}, \mathbf{H}) \Leftrightarrow d(\mathbf{A}, \mathbf{B}) \leq d(\mathbf{C}, \mathbf{H})$. By substituting the ROOTED distance, it turns out that we want to show that

$$\sqrt{\sum_{i=1}^n \sum_{j=1}^n (\sqrt{s_{\mathbf{A}, ij}} - \sqrt{s_{\mathbf{B}, ij}})^2} \leq \sqrt{\sum_{i=1}^n \sum_{j=1}^n (\sqrt{s_{\mathbf{C}, ij}} - \sqrt{s_{\mathbf{H}, ij}})^2},$$

where s_{ij} are the elements of the corresponding affinity matrix \mathbf{S} . These are defined for \mathbf{A} by expressing the matrix inversion in Eq. (1) using a power series and ignoring the terms of greater than second power:

$$\vec{s}_i = [\mathbf{I} + (\epsilon \mathbf{A} - \epsilon^2 \mathbf{D}_{\mathbf{A}}) + (\epsilon \mathbf{A} - \epsilon^2 \mathbf{D}_{\mathbf{A}})^2 + \dots] \vec{e}_i \Rightarrow \mathbf{S}_{\mathbf{A}} = \mathbf{I} + \epsilon \mathbf{A} + \epsilon^2 \mathbf{A}^2 - \epsilon^2 \mathbf{D}_{\mathbf{A}},$$

⁴We use \mathbf{H} instead of \mathbf{D} to distinguish the adjacency matrix of the graph from the diagonal matrix of degrees which is normally defined as \mathbf{D} .

where the index (e.g., \mathbf{A}) denotes the graph each matrix corresponds to.

The proof is very technical and far from easy. Thus, instead we show a representative set of simulations that suggest that the submodularity property is satisfied by DELTACON. Specifically, we start from a complete graph of size n , $G_0 = K_n$, and randomly pick an edge (i_0, j_0) . Then, we generate a series of graphs, G_t , derived by G_{t-1} by removing one new edge (i_t, j_t) . Note that (i_t, j_t) cannot be the initially chosen edge (i_0, j_0) . For every derived graph, we compute the ROOTED distance between itself and the same graph without the edge (i_0, j_0) . What we expect to see is that the distance decreases as the number of edges in the graph increases. In other words, the distance between a sparse graph and the same graph without (i_0, j_0) is bigger than the distance between a denser graph and the same graph missing the edge (i_0, j_0) . The opposite holds for the DELTACON similarity measure, but in the proofs we use the distance since that function is mathematically easier to manipulate than the similarity function. In Fig. 2, we plot, for different graph sizes n , the ROOTED distance as a function of the edges in the graph (from left to right the graph becomes denser, and tends to the complete graph K_n).

P3. [Weight Awareness] In weighted graphs, the bigger the weight of the removed edge is, the greater the impact on the similarity measure should be.

LEMMA 4.

Assume that G_A is a graph with adjacency matrix \mathbf{A} and elements $a_{ij} \geq 0$. Also, let G_B be a graph with adjacency matrix \mathbf{B} and elements

$$b_{ij} = \begin{cases} a_{ij} + k & \text{if } (i, j) = (i_0, j_0) \text{ or } (i, j) = (j_0, i_0) \\ a_{ij} & \text{otherwise} \end{cases}$$

where k is a positive integer ($k \geq 1$). Then, it holds that $(\mathbf{S}_B)_{ij} \geq (\mathbf{S}_A)_{ij}$.

PROOF OF LEMMA 4. From eq. (1), by expressing the matrix inversion using a power series and ignoring the terms of greater than second power, we obtain the solution:

$$\vec{s}_i = [\mathbf{I} + (\epsilon\mathbf{A} - \epsilon^2\mathbf{D}) + (\epsilon\mathbf{A} - \epsilon^2\mathbf{D})^2 + \dots]\vec{e}_i \Rightarrow \vec{s}_i \approx [\mathbf{I} + \epsilon\mathbf{A} + \epsilon^2(\mathbf{A}^2 - \mathbf{D})]\vec{e}_i,$$

or equivalently

$$\mathbf{S}_A = \mathbf{I} + \epsilon\mathbf{A} + \epsilon^2\mathbf{A}^2 - \epsilon^2\mathbf{D}_A$$

and

$$\mathbf{S}_B = \mathbf{I} + \epsilon\mathbf{B} + \epsilon^2\mathbf{B}^2 - \epsilon^2\mathbf{D}_B.$$

Note that

$$\mathbf{S}_B - \mathbf{S}_A = \epsilon(\mathbf{B} - \mathbf{A}) + \epsilon^2(\mathbf{B}^2 - \mathbf{A}^2) - \epsilon^2(\mathbf{D}_B - \mathbf{D}_A), \quad (4)$$

where

$$(\mathbf{B} - \mathbf{A})_{ij} = \begin{cases} k & \text{if } (i, j) = (i_0, j_0) \text{ or } (i, j) = (j_0, i_0) \\ 0 & \text{otherwise} \end{cases}$$

and

$$(\mathbf{D}_B - \mathbf{D}_A)_{ij} = \begin{cases} k & \text{if } (i, j) = (i_0, j_0) \text{ or } (i, j) = (j_0, i_0) \\ 0 & \text{otherwise.} \end{cases}$$

By applying basic algebra, it can be shown that

$$(\mathbf{B}^2)_{ij} \begin{cases} = (\mathbf{A}^2)_{ij} + 2k \cdot a_{i_0 j_0} + k^2 & \text{if } (i, j) = (i_0, i_0) \text{ or } (i, j) = (j_0, j_0) \\ \geq (\mathbf{A}^2)_{ij} & \text{otherwise} \end{cases}$$

since $b_{ij} \geq a_{ij} \geq 0$ for all i, j by definition. Next, observe that for Eq. 4, we have 3 cases:

— For all $(i, j) \neq (i_0, i_0)$ and (j_0, j_0) , we have

$$(\mathbf{S}_B - \mathbf{S}_A)_{ij} = \epsilon(\mathbf{B} - \mathbf{A})_{ij} + \epsilon^2(\mathbf{B}^2 - \mathbf{A}^2)_{ij} \geq 0$$

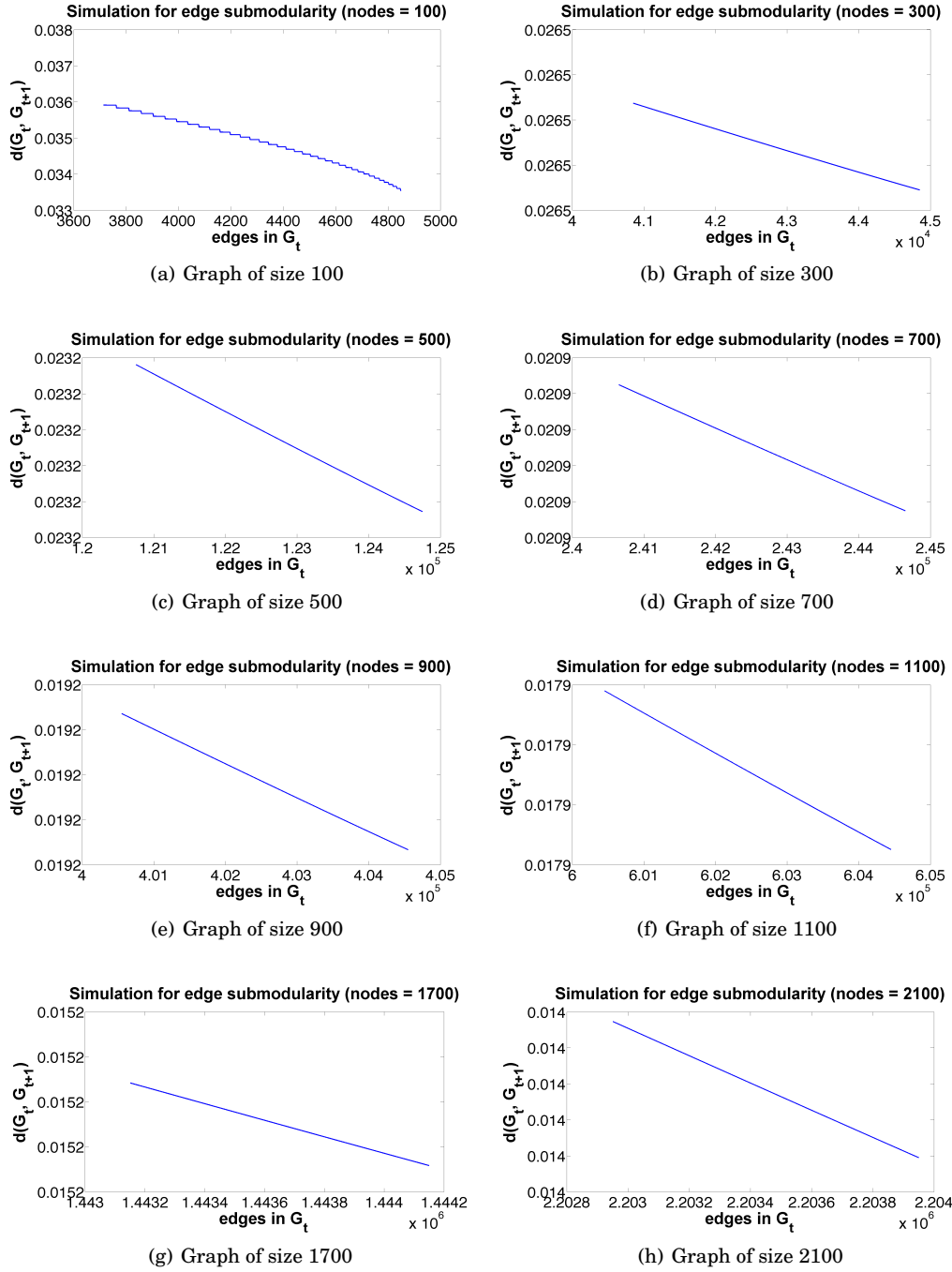


Fig. 2: Extensive simulations for different graph sizes indicate that DELTACON satisfies the edge-submodularity property. The distance between two graphs that differ only in the edge (i_0, j_0) is a decreasing function of the number of edges in the original graph G_t . Or reversely, the their similarity is an increasing function of the number of edges in the original graph G_t .

— For $(i, j) = (i_0, i_0)$ we have

$$(\mathbf{S}_B - \mathbf{S}_A)_{i_0 i_0} = \epsilon^2 k(2k \cdot a_{i_0 j_0} + k - 1) \geq 0$$

— For $(i, j) = (j_0, j_0)$ we have

$$(\mathbf{S}_B - \mathbf{S}_A)_{j_0 j_0} = \epsilon^2 k(2k \cdot a_{i_0 j_0} + k - 1) \geq 0$$

Hence, for all (i, j) it holds that $(\mathbf{S}_B)_{ij} \geq (\mathbf{S}_A)_{ij}$. \square

Next we will use the lemma to prove the weight awareness property.

PROOF OF PROPERTY P3. We formalize the weight awareness property in the following way. Let \mathbf{A} be the adjacency matrix of a weighted, undirected graph, with elements a_{ij} . Then, \mathbf{B} is equal \mathbf{A} but with a bigger weight for the edge (i_0, j_0) , or more formally:

$$b_{ij} = \begin{cases} a_{ij} + k & \text{if } (i, j) = (i_0, j_0) \text{ or } (i, j) = (j_0, i_0) \\ a_{ij} & \text{otherwise} \end{cases}$$

Let also \mathbf{C} be the adjacency matrix of another graph with the same entries as \mathbf{A} except for c_{i_0, j_0} , which is bigger than b_{i_0, j_0} :

$$c_{ij} = \begin{cases} a_{ij} + k' & \text{if } (i, j) = (i_0, j_0) \text{ or } (i, j) = (j_0, i_0) \\ a_{ij} & \text{otherwise} \end{cases}$$

where $k' > k$ is an integer. To prove the property, it suffices to show that $\text{sim}(\mathbf{A}, \mathbf{B}) \geq \text{sim}(\mathbf{A}, \mathbf{C}) \Leftrightarrow d(\mathbf{A}, \mathbf{B}) \leq d(\mathbf{A}, \mathbf{C})$. Notice that this formal definition includes the case of removing an edge by assuming that $a_{i_0, j_0} = 0$ for matrix \mathbf{A} .

We can write the difference of the squares of the ROOTED distances as:

$$\begin{aligned} d^2(\mathbf{A}, \mathbf{B}) - d^2(\mathbf{A}, \mathbf{C}) &= \sum_{i=1}^n \sum_{j=1}^n (\sqrt{s_{\mathbf{A}, ij}} - \sqrt{s_{\mathbf{B}, ij}})^2 - \sum_{i=1}^n \sum_{j=1}^n (\sqrt{s_{\mathbf{A}, ij}} - \sqrt{s_{\mathbf{C}, ij}})^2 \\ &= \sum_{i=1}^n \sum_{j=1}^n (\sqrt{s_{\mathbf{C}, ij}} - \sqrt{s_{\mathbf{B}, ij}})(2\sqrt{s_{\mathbf{A}, ij}} - \sqrt{s_{\mathbf{B}, ij}} - \sqrt{s_{\mathbf{C}, ij}}) < 0 \end{aligned}$$

because $(\mathbf{S}_B)_{ij} \geq (\mathbf{S}_A)_{ij}$, $(\mathbf{S}_C)_{ij} \geq (\mathbf{S}_A)_{ij}$, and $(\mathbf{S}_C)_{ij} \geq (\mathbf{S}_B)_{ij}$ for all i, j by the construction of the matrices \mathbf{A} , \mathbf{B} , and \mathbf{C} , and Lemma 4. \square

In Section 5, we show experimentally that not only DELTACON satisfies the properties, but also that other similarity and distance methods fail in one or more test cases.

4. DELTACON-ATTR: ADDING NODE AND EDGE ATTRIBUTION

Thus far, we have broached the intuition and decisions behind developing DELTACON for calculating graph similarity. However, we argue that computing this metric is only half the battle in the wider realm of change detection and graph understanding. Equally important is finding out *why* the graph changed the way it did. One way of doing this is by attributing the changes to nodes and/or edges.

Equipped with this information, we can draw conclusions with respect to how certain changes impact the graph connectivity and apply this understanding in a domain-specific context to assign blame, as well as instrument measures to prevent such changes in the future. Additionally, such a feature can be used to measure changes which have not yet happened in order to find information about which nodes and/or edges are most important for preserving or destroying connectivity. In this section, we will discuss our extension of DELTACON, called DELTACON-ATTR, which enables node- and edge-level attribution for this very purpose.

4.1. Algorithm Description

Node Attribution. Our first goal is to find the nodes which are mostly responsible for the difference between the input graphs. Let the affinity matrices \mathbf{S}'_1 and \mathbf{S}'_2 be precomputed. Then, the steps of our node attribution algorithm (Algorithm 3) can be summarized to:

Step 1. Intuitively, we compute the difference between the affinity of node v to the node groups in graph A and the affinity of node v to the node groups in graph A_2 . To that end, we use the same distance, ROOTED, that we applied to find the similarity between the whole graphs.

Given that the v th row vector ($v \leq n$) of S'_1 and S'_2 reflects the affinity of node v to the remainder of the graph, the ROOTED distance between the two vectors provides a measure of the extent to which that node is a *culprit* for change — we refer to this measure as the *impact* of a node. Thus, culprits with comparatively high impact are ones that are most responsible for change between graphs.

More formally, we quantify the contribution of each node to the graph changes by taking the ROOTED distance between each corresponding pair of row vectors in S'_1 and S'_2 as \vec{w}_v for $v = 1, \dots, n$ per Eq. (5).

$$\vec{w}_v = \text{ROOTED}(S'_{1,v}, S'_{2,v}) = \sqrt{\sum_{j=1}^g (\sqrt{s'_{1,vj}} - \sqrt{s'_{2,vj}})^2}. \quad (5)$$

Step 2. We sort the scores in the $n \times 1$ node impact vector \vec{w} in descending order and report the most important scores and their corresponding nodes.

ALGORITHM 3: DELTACON-ATTR Node Attribution

```

INPUT: affinity matrices  $S'_1, S'_2$ 
 $\tau = 0.8$  // culprit reporting threshold  $\tau$  defaulted to 0.8 ( $0 \leq \tau \leq 1$ )
for  $v = 1 \rightarrow n$  do
     $\vec{w}_v = \text{ROOTED}(S'_{1,v}, S'_{2,v})$ 
end for
 $[\vec{w}_{\text{sorted}}, \vec{w}_{\text{sortedIndex}}] = \text{sortrows}(\vec{w}, 1, \text{'descend'})$  // sort rows of vector  $\vec{w}$  on column index 1
// (node impact score) by descending value

runningTotal = 0
index = 1
sumScores = sum( $\vec{w}_{\text{sorted}}$ )
while runningTotal  $\leq \tau$  do
    runningTotal = runningTotal +  $\frac{\vec{w}_{\text{sorted}, \text{index}}}{\text{sumScores}}$ 
    index = index + 1
end while
return  $[\vec{w}_{\text{sorted}, 1 \dots \text{index}}, \vec{w}_{\text{sortedIndex}, 1 \dots \text{index}}]$ 

```

By default, we report culprits responsible for the top 80% of changes using a similar line of reasoning as that behind Fukunaga's heuristic [Fukunaga 1990]. In practice, we find that the notion of a skewed impact distribution holds (though the law of factor sparsity holds, the distribution need not be 80-20).

Edge Attribution. Complementarily to the node attribution approach, we also developed an edge attribution method which ranks edge changes (additions and deletions) with respect to the graph changes. The steps of our edge attribution algorithm (Algorithm 4) are:

Step 1. We assign each changed edge incident to at least one node in the culprit set of interest an impact score equal to the sum of impact scores for the nodes which it connects or disconnects.

Our goal here is to assign edges impact according to the degree that they affect the nodes that they touch. Since even the removal or addition of a single edge does not necessarily impact both incident nodes equally, we choose the sum of both nodes' scores as the edge impact metric. Thus, our algorithm will rank edges which touch two nodes of moderately high impact more importantly than edges which touch one node of high impact but another of low impact.

Step 2. We sort the edge impact scores in descending order and report the edges in order of importance.

Analysis of changed edges can reveal important discrepancies from baseline behavior. Specifically, a large number of added edges or removed edges with individually low impact is indicative of star formation or destruction, whereas one or a few added or removed edges with individually high impact is indicative of community expansion or reduction via addition or removal of certain key bridge edges.

ALGORITHM 4: DELTACON-ATTR Edge Attribution

INPUT: adjacency matrices A_1, A_2 , culprit set of interest $\vec{w}_{sortedIndex, 1 \dots index}$ and node impact scores \vec{w}

```

for  $v = 1 \rightarrow \text{length}(\vec{w}_{sortedIndex, 1 \dots index})$  do
   $srcNode = \vec{w}_{sortedIndex, v}$ 
   $\vec{r} = A_{2,v} - A_{1,v}$ 
  for  $k = 1 \rightarrow n$  do
     $destNode = k$ 
    if  $\vec{r}_k = 1$  then
       $edgeScore = \vec{w}_{srcNode} + \vec{w}_{destNode}$ 
      append row  $[srcNode, destNode, '+', edgeScore]$  to  $E$ 
    end if
    if  $\vec{r}_k = -1$  then
       $edgeScore = \vec{w}_{srcNode} + \vec{w}_{destNode}$ 
      append row  $[srcNode, destNode, '-', edgeScore]$  to  $E$ 
    end if
  end for
end for
 $E_{sorted} = \text{sortrows}(E, 4, 'descend')$  // sort rows of matrix  $E$  on column index 4 (edge impact
// score) by descending value

return  $E_{sorted}$ 

```

4.2. Scalability Analysis

Given precomputed S'_1 and S'_2 (precomputation is assumed since attribution can only be conducted after similarity computation), the node attribution component of DELTACON-ATTR is loglinear on the number of nodes, since n influence scores need to be sorted in descending order. In more detail, the cost of computing the impact scores for nodes is linear on the number of nodes and groups, but is dominated by the sorting cost given that $g \ll \log(n)$ in general.

With the same assumptions with respect to precomputed results, the edge attribution portion of DELTACON-ATTR is also loglinear, but on the sum of edge counts, since $m_1 + m_2$ total possible changed edges need to be sorted. In practice, the number of edges needing to be sorted should be far smaller, as we need only concern ourselves with edges which are incident to nodes in the culprit set of interest. Specifically, the cost of computing impact scores for edges is linear on the number of nodes in the culprit set k and the number of changed edges, but is again dominated by the sorting cost given that $k \ll \log(m_1 + m_2)$ in general.

5. EXPERIMENTS

We conduct several experiments on synthetic (Fig. 3), as well as real data (undirected, unweighted graphs, unless stated otherwise - see Table VI) to answer the following questions:

- Q1.** Does DELTACON agree with our intuition and satisfy the axioms/properties? Where do other methods fail?
- Q2.** Is DELTACON scalable and able to compare large-scale graphs?
- Q3.** How sensitive is it to the number of node groups?

We implemented the code in Matlab and ran the experiments on AMD Opteron Processor 854 @3GHz, RAM 32GB. The selection of parameters in Eq. (1) follows the lines of the paper [Koutra et al. 2011]; all the parameters are chosen so that the system converges.

5.1. Intuitiveness of DELTACON

To answer Q1, for the first 3 properties (P1-P3), we conduct experiments on small graphs of 5 to 100 nodes and classic topologies (cliques, stars, circles, paths, barbell and wheel-barbell graphs, and “lollipops” shown in Fig. 3), since people can argue about their similarities. For the name conventions see Table 2. For our method we used 5 groups (g), but the results are similar for other choices of the parameter. In addition to the synthetic graphs, for the informal property (IP), we use real networks with up to 11 million edges (Table VI, page 20).

We compare our method, DELTACON, to the 6 best state-of-the-art similarity measures that apply to our setting:

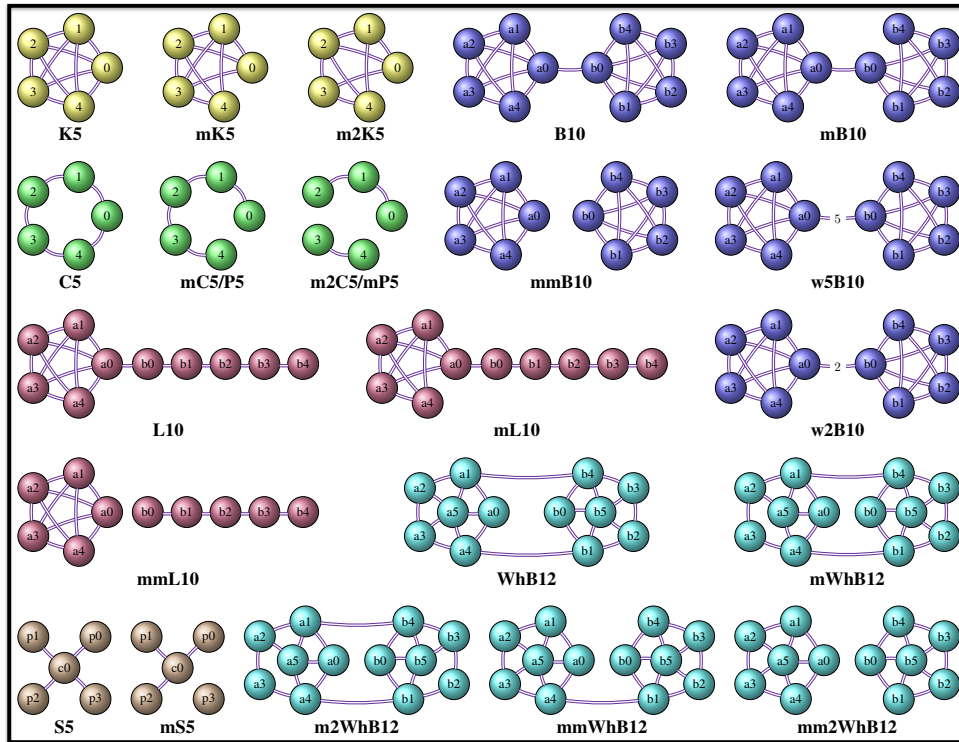


Fig. 3: Small, synthetic graphs used in DELTACON experimental analysis – K : clique, C : cycle, P : path, S : star, B : barbell, L : lollipop, and WhB : wheel-barbell.

Table II: Name conventions for synthetic graphs. Missing number after the prefix implies $X = 1$.

Symbol	Meaning	Symbol	Meaning
K_n	clique of size n	B_n	barbell of size n
P_n	path of size n	WhB_n	wheel barbell of size n
C_n	cycle of size n	m_x	missing X edges
S_n	star of size n	mm_x	missing X “bridge” edges
L_n	lollipop of size n	w	weight of “bridge” edge

- (1) **Vertex/Edge Overlap (VEO)**: In [Papadimitriou et al. 2008], the VEO similarity between two graphs $G_1(\mathcal{V}_1, \mathcal{E}_1)$ and $G_2(\mathcal{V}_2, \mathcal{E}_2)$ is defined as:

$$\text{sim}_{VEO}(G_1, G_2) = 2 \frac{|\mathcal{E}_1 \cap \mathcal{E}_2| + |\mathcal{V}_1 \cap \mathcal{V}_2|}{|\mathcal{E}_1| + |\mathcal{E}_2| + |\mathcal{V}_1| + |\mathcal{V}_2|}.$$

- (2) **Graph Edit Distance (GED)**: GED has quadratic complexity in general, but it is linear on the number of nodes and edges when only insertions and deletions are allowed [Bunke et al. 2006].

$$\begin{aligned} \text{sim}_{GED}(G_1, G_2) = & |\mathcal{V}_1| + |\mathcal{V}_2| - 2|\mathcal{V}_1 \cap \mathcal{V}_2| \\ & + |\mathcal{E}_1| + |\mathcal{E}_2| - 2|\mathcal{E}_1 \cap \mathcal{E}_2|. \end{aligned}$$

For $\mathcal{V}_1 = \mathcal{V}_2$ and unweighted graphs, sim_{GED} is equivalent to the Hamming distance (HD) defined as $\text{HD}(\mathbf{A}_1, \mathbf{A}_2) = \text{sum}(\mathbf{A}_1 \text{ XOR } \mathbf{A}_2)$.

- (3) **Signature Similarity (SS)**: This is the best performing similarity measure studied in [Papadimitriou et al. 2008]. It starts from node and edge features, and by applying the SimHash algorithm (random projection based method), projects the features to a small dimension feature space, which is called *signature*. The similarity between the graphs is defined as the similarity between their signatures.
- (4) The last 3 methods are variations of the well-studied spectral method “ λ -distance” ([Bunke et al. 2006], [Peabody 2003], [Wilson and Zhu 2008]). Let $\{\lambda_{1i}\}_{i=1}^{|\mathcal{V}_1|}$ and $\{\lambda_{2i}\}_{i=1}^{|\mathcal{V}_2|}$ be the eigenvalues of the matrices that represent G_1 and G_2 . Then, λ -distance is given by

$$d_\lambda(G_1, G_2) = \sqrt{\sum_{i=1}^k (\lambda_{1i} - \lambda_{2i})^2},$$

where k is $\max(|\mathcal{V}_1|, |\mathcal{V}_2|)$ (padding is required for the smallest vector of eigenvalues). The variations of the method are based on three different matrix representations of the graphs: adjacency (λ -D Adj.), laplacian (λ -D Lap.) and normalized laplacian matrix (λ -D N.L.).

The results for the first 3 properties are presented in the form of tables III-IV. For property P1 we compare the graphs (A,B) and (A,C) and report the difference between the pairwise similarities/distances of our proposed methods and the 6 state-of-the-art methods. We have arranged the pairs of graphs in such way that (A,B) are more similar than (A,C). Therefore, table entries that are non-positive mean that the corresponding method does not satisfy the property. Similarly, for properties P2 and P3, we compare the graphs (A,B) and (C,D) and report the difference in their pairwise similarity/distance scores.

P1. Edge Importance. “Edges whose removal creates disconnected components are more important than other edges whose absence does not affect the graph connectivity. The more important an edge is, the more it should affect the similarity or distance measure.”

For this experiment we use the barbell, “wheel barbell” and “lollipop” graphs depicted in Fig. 3, since it is easy to argue about the importance of the individual edges. The idea is that edges in a highly connected component (e.g. clique, wheel) are not very important from the information flow viewpoint, while edges that *connect* (almost uniquely) dense components play a significant role in the connectivity of the graph and the information flow. The importance of the “bridge” edge depends on the size of the components that it connects; the bigger the components the more important is the role of the edge.

OBSERVATION 2. Only DELTACON succeeds in distinguishing the importance of the edges (P1) w.r.t. connectivity, while all the other methods fail at least once (Table III).

P2. “Edge-Submodularity”. “Let $A(\mathcal{V}, \mathcal{E}_1)$ and $B(\mathcal{V}, \mathcal{E}_2)$ be two unweighted graphs with the same node set, and $|\mathcal{E}_1| > |\mathcal{E}_2|$ edges. Also, assume that $m_x A(\mathcal{V}, \mathcal{E}_1)$ and $m_x B(\mathcal{V}, \mathcal{E}_2)$ are the respective derived graphs after removing x edges. We expect that $\text{sim}(A, m_x A) > \text{sim}(B, m_x B)$, since the fewer the edges in a constant-sized graph, the more “important” they are.”

Table III: “Edge Importance” (P1). Highlighted entries violate P1.

GRAPHS				DC ₀	DC	VEO	SS	GED (XOR)	λ -D ADJ.	λ -D LAP.	λ -D N.L.
A	B	C		$\Delta s = \text{sim}(A, B) - \text{sim}(A, C)$				$\Delta d = d(A, C) - d(A, B)$			
B10	mB10	mmB10		0.07	0.04	0	-10^{-5}	0	0.21	-0.27	2.14
L10	mL10	mmL10		0.04	0.02	0	10^{-5}	0	-0.30	-0.43	-8.23
WhB10	mWhB10	mmWhB10		0.03	0.01	0	-10^{-5}	0	0.22	0.18	-0.41
WhB10	m2WhB10	mm2WhB10		0.07	0.04	0	-10^{-5}	0	0.59	0.41	0.87

Table IV: “Edge-Submodularity” (P2). Highlighted entries violate P2.

GRAPHS				DC ₀	DC	VEO	SS	GED (XOR)	λ -D ADJ.	λ -D LAP.	λ -D N.L.
A	B	C	D	$\Delta s = \text{sim}(A, B) - \text{sim}(C, D)$				$\Delta d = d(C, D) - d(A, B)$			
K5	mK5	C5	mC5	0.03	0.03	0.02	10^{-5}	0	-0.24	-0.59	-7.77
C5	mC5	P5	mP5	0.03	0.01	0.01	-10^{-5}	0	-0.55	-0.39	-0.20
K ₁₀₀	mK ₁₀₀	C ₁₀₀	mC ₁₀₀	0.03	0.02	0.002	10^{-5}	0	-1.16	-1.69	-311
C ₁₀₀	mC ₁₀₀	P ₁₀₀	mP ₁₀₀	10^{-4}	0.01	10^{-5}	-10^{-5}	0	-0.08	-0.06	-0.08
K ₁₀₀	m10K ₁₀₀	C ₁₀₀	m10C ₁₀₀	0.10	0.08	0.02	10^{-5}	0	-3.48	-4.52	-1089
C ₁₀₀	m10C ₁₀₀	P ₁₀₀	m10P ₁₀₀	0.001	0.001	10^{-5}	0	0	-0.03	0.01	0.31

Table V: “Weight Awareness” (P3). Highlighted entries violate P3.

GRAPHS				DC ₀	DC	VEO	SS	GED (XOR)	λ -D ADJ.	λ -D LAP.	λ -D N.L.
A	B	C	D	$\Delta s = \text{sim}(A, B) - \text{sim}(C, D)$				$\Delta d = d(C, D) - d(A, B)$			
B10	mB10	B10	w5B10	0.09	0.08	-0.02	10^{-5}	-1	3.67	5.61	84.44
mmB10	B10	mmB10	w5B10	0.10	0.10	0	10^{-4}	0	4.57	7.60	95.61
B10	mB10	w5B10	w2B10	0.06	0.06	-0.02	10^{-5}	-1	2.55	3.77	66.71
w5B10	w2B10	w5B10	mmB10	0.10	0.07	0.02	10^{-5}	1	2.23	3.55	31.04
w5B10	w2B10	w5B10	B10	0.03	0.02	0	10^{-5}	0	1.12	1.84	17.73

Tables III-V: DELTA_{CON}₀ and DELTA_{CON} (in bold) obey all the formal required properties (P1-P3). Each row of the tables corresponds to a comparison between the similarities (or distances) of two pairs of graphs; pairs (A,B) and (A,C) for property (P1); and pairs (A,B) and (C,D) for (P2) and (P3). Non-positive values of $\Delta s = \text{sim}(A, B) - \text{sim}(C, D)$ and $\Delta d = d(C, D) - d(A, B)$ for similarity and distance methods respectively, are highlighted and mean violation of the property of interest.

The results for different graph topologies and 1 or 10 removed edges (prefixes ‘m’ and ‘m10’ respectively) are given compactly in Table IV. Recall that non-positive values denote violation of the “edge-submodularity” property.

OBSERVATION 3. Only DELTA_{CON} complies to the “edge-submodularity” property (P2) in all cases examined.

P3. Weight Awareness. “The absence of an edge of big weight is more important than the absence of a smaller weighted edge; this should be reflected in the similarity measure.”

The weight of an edge defines the strength of the connection between two nodes, and, in this sense, can be viewed as a feature that relates to the importance of the edge in the graph. For this property, we study the weighted versions of the barbell graph, where we assume that all the edges except the “bridge” have unit weight.

OBSERVATION 4. All the methods are weight-aware ($P3$), except VEO and GED, which compute just the overlap in edges and vertices between the graphs (Table V).

IP. Focus Awareness. At this point, all the competing methods have failed in satisfying at least one of the formal desired properties. To test whether DELTACON satisfies our *informal* property, i.e., it is able to distinguish the extent of a change in a graph, we analyze real datasets with up to 11 million edges (Table VI) for two different types of changes. For each graph we create corrupted instances by removing: (i) edges from the original graph randomly, and (ii) the same number of edges in a targeted way (we randomly choose nodes and remove all their edges, until we have removed the appropriate fraction of edges).

For this property, we study 8 real networks: Email EU and Enron Emails, Facebook wall and Facebook links, Google and Stanford web, Berkeley/Stanford web and AS Skitter. In Fig. 4, we give for different levels of corruption (10%, 20%, ..., 80%) the DELTACON similarity score between the original graph and the corrupted graph when the edge removal is random (solid line), and the DELTACON score when the edge removal is targeted (dashed line).

OBSERVATION 5.

- “*Targeted changes hurt more.*” DELTACON is focus-aware (IP). Removal of edges in a targeted way leads to smaller similarity of the derived graph to the original one than removal of the same number of edges in a random way.
- “*More changes: random \approx targeted.*” In Fig. 4, as the fraction of removed edges increases, the similarity score for random changes (solid lines) tends to the similarity score for targeted changes (dashed lines).

This is expected, because the random and targeted edge removal tend to be equivalent when a significant fraction of edges is deleted.

In Fig. 5, we give the similarity score as a function of the percent of the removed edges. Specifically, the x axis corresponds to the percentage of edges that have been removed from the original graph, and the y axis gives the similarity score. As before, each point maps to the similarity score between the original graph and the corresponding corrupted graph.

Table VI: Large Real and Synthetic Datasets

Name	Nodes	Edges	Description
Brain Graphs Small [OCP 2014]	70	800-1,208	connectome
Enron Email [Klimt and Yang 2004]	36,692	367,662	who-emails-whom
Facebook wall [Viswanath et al. 2009]	45,813	183,412	who-posts-to-whom
Facebook links [Viswanath et al. 2009]	63,731	817,090	friend-to-friend
Epinions [Guha et al. 2004]	131,828	841,372	who-trusts-whom
Email EU [Leskovec et al. 2007]	265,214	420,045	who-sent-to-whom
Web Notre Dame [Leskovec 2013]	325,729	1,497,134	site-to-site
Web Stanford [Leskovec 2013]	281,903	2,312,497	site-to-site
Web Google [Leskovec 2013]	875,714	5,105,039	site-to-site
Web Berk/Stan [Leskovec 2013]	685,230	7,600,595	site-to-site
AS skitter [Leskovec et al. 2007]	1,696,415	11,095,298	p2p links
Brain Graphs Big [OCP 2014]	16,777,216	49,361,130-90,492,237	connectome
Kronecker 1	6,561	65,536	synthetic
Kronecker 2	19,683	262,144	synthetic
Kronecker 3	59,049	1,048,576	synthetic
Kronecker 4	177,147	4,194,304	synthetic
Kronecker 5	531,441	16,777,216	synthetic
Kronecker 6	1,594,323	67,108,864	synthetic

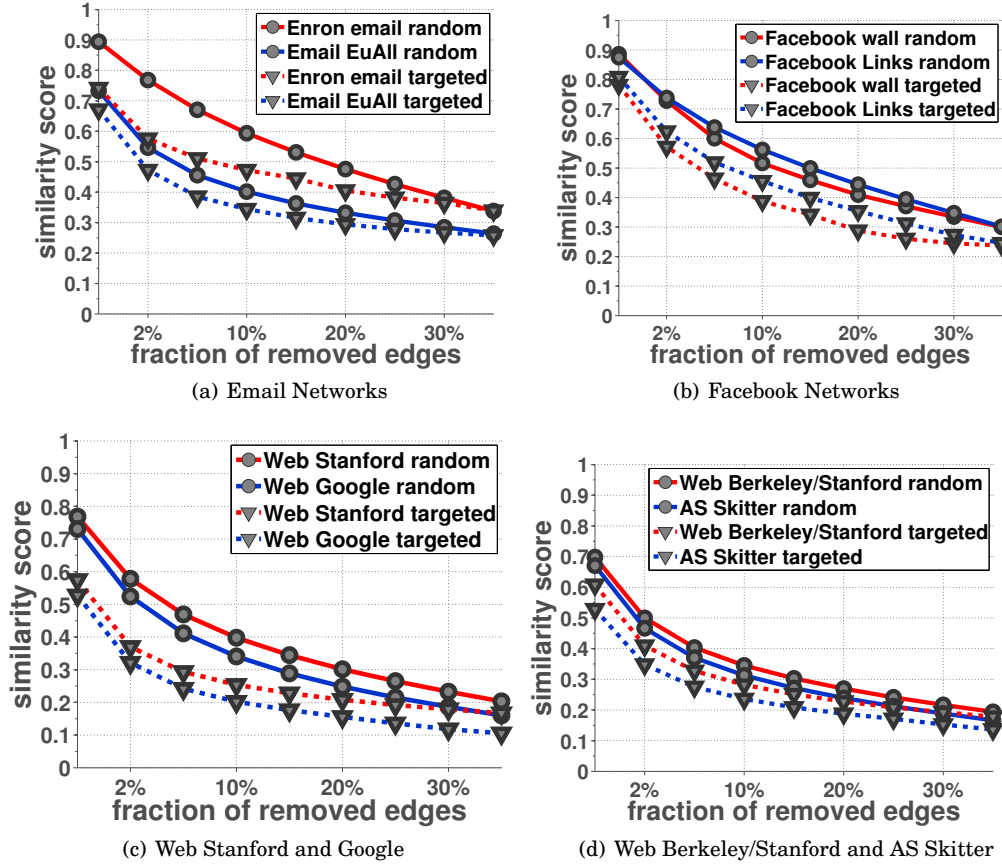


Fig. 4: DELTACON is “focus-aware” (IP) – targeted changes hurt more than random. Plots of DELTACON similarity scores for random -solid lines- and targeted -dashed lines- changes (y axis) vs. the fraction of removed edges in the “corrupted” versions of the original graphs (x axis). Notice that the dashed lines are always below the solid lines of same color. DELTACON agrees with intuition: the more a graph changes (i.e., the number of removed edges increases), the smaller is its similarity to the original graph.

OBSERVATION 6. “More changes hurt more.” The higher the corruption of the original graph, the smaller the DELTACON similarity between the derived and the original graph is. In Fig. 5, we observe that as the percentage of removed edges increases, the similarity to the original graph decreases consistently for a variety of real graphs.

General Remarks. All in all, the baseline methods have several non-desirable properties. The spectral methods, as well as SS fail to comply to the “edge importance” (P1) and “edge sub-modularity” (P2) properties. Moreover, λ -distance has high computational cost when the whole graph spectrum is computed, cannot distinguish the differences between co-spectral graphs, and sometimes small changes lead to big differences in the graph spectra. As far as VEO and GED are concerned, they are oblivious on significant structural properties of the graphs; thus, despite their straightforwardness and fast computation, they fail to discern various changes in the graphs. On the other hand, DELTACON gives tangible similarity scores and conforms to all the desired properties.

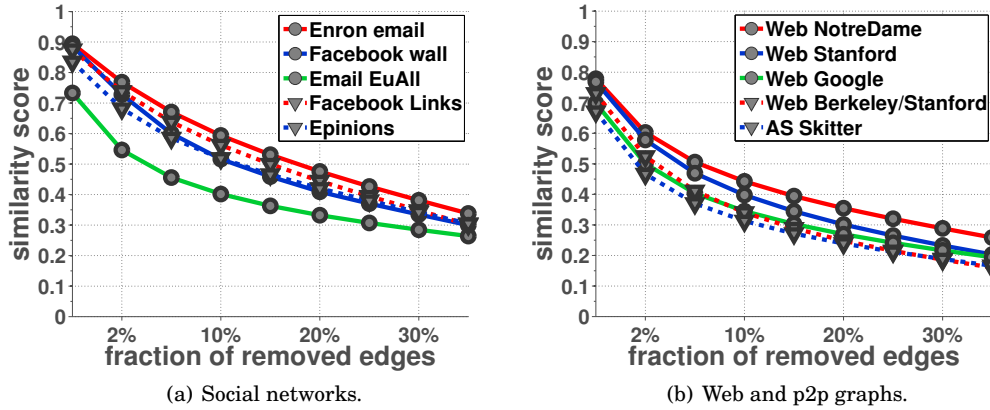


Fig. 5: DELTACON agrees with intuition: the more a graph changes (i.e., the number of removed edges increases), the smaller is its similarity to the original graph.

5.2. Intuitiveness of DELTACON-ATTR

In addition to evaluating the intuitiveness of DELTACON_0 and DELTACON , we extensively tested $\text{DELTA}\text{CON-ATTR}$ on a number of synthetically created and modified graphs. We found that $\text{DELTA}\text{CON-ATTR}$ reflects the desired properties P1-P4. Several of the conducted experiments are depicted in Fig. 6, where we also annotate the node and edge culprits that explain the main differences between the compared graphs.

The first pair of graphs corresponds to the lollipop graph, $L10$, and the lollipop variant, $mL10 \cap mmL10$, that is missing one edge from the clique, as well as a bridge edge. Nodes $b0$, $a0$ and $a4$ are considered the top culprits for the difference in the graphs. Moreover, $b0$ is ranked more responsible for the change than $a0$, since $b0$ lost connectivity to a more strongly connected component than $a0$.

The second pair consists of a 5-node clique and the same clique with one missing edge, $(0,4)$. $\text{DELTA}\text{CON-ATTR}$ considers nodes 0 and 4 top culprits, with equal rank, due to equivalent loss in connectivity. Edge $(0,4)$ is ranked top, and is essentially the only changed edge.

The third pair is the barbell example, where we compare a barbell graph of 10 nodes to the barbell graph that is missing both an edge from a clique, $(b0, b4)$, as well as the bridge edge, $(a0, b0)$. As expected, $\text{DELTA}\text{CON-ATTR}$ finds $b0$, $a0$ and $b4$ as top culprits, where $b0$ is ranked higher than

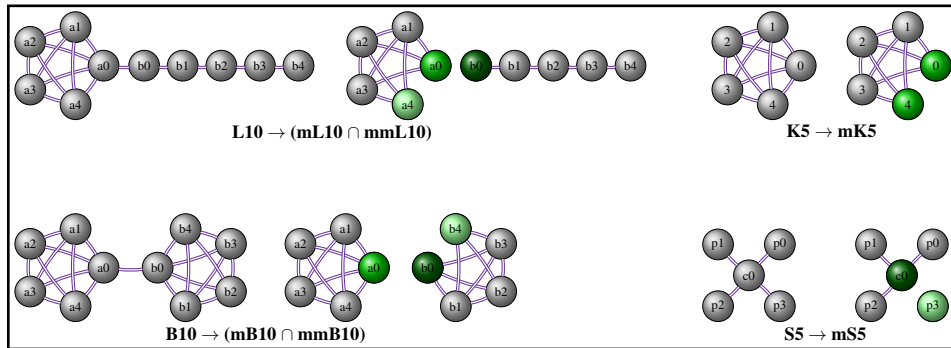


Fig. 6: $\text{DELTA}\text{CON-ATTR}$ respects properties P1-P4. Each pair of graphs depicts the before and after versions of a synthetically modified graph. Nodes marked green are identified as the main culprits for the difference between the graphs. Darker shade corresponds to higher rank in the list of culprits.

$a0$, since $b0$ lost connectivity to both nodes $a0$ and $b4$, whereas $a0$ lost direct connectivity to only $b0$.

The last pair that we present here consists of a 5-node star graph, and the same graph missing the edge $(c0, p3)$. DELTACON-ATTR considers $p3$ and $c0$ top culprits, with $p3$ ranking higher than $c0$ as the edge removal caused loss of connectivity to all the peripheral nodes of the star, $p0, p1$ and $p2$.

As we observe, in all the synthetic and easily controlled examples, the ranking of the culprit nodes and edges that DELTACON-ATTR finds agrees with intuition.

5.3. Scalability of DELTACON

In Section 2 we demonstrated that DELTACON is linear on the number of edges, and here we show that this also holds in practice. We ran DELTACON on Kronecker graphs (Table VI), which are known [Leskovec et al. 2005] to share many properties with real graphs.

OBSERVATION 7. As shown in Fig. 7, DELTACON scales linearly with the number of edges in the largest input graph.

Notice that the algorithm can be trivially parallelized by finding the node affinity scores of the two graphs in parallel instead of sequential. Moreover, for each graph the computation of the similarity scores of the nodes to each of the g groups can be parallelized. However, the runtime of our experiments refer to the sequential implementation. The amount of time taken for DELTACON-ATTR is trivial even for large graphs, given that the necessary affinity matrices are in memory already from the DELTACON similarity computation. Specifically, node and edge attribution are loglinear on nodes and edges respectively given that sorting is unavoidable for the task of ranking.

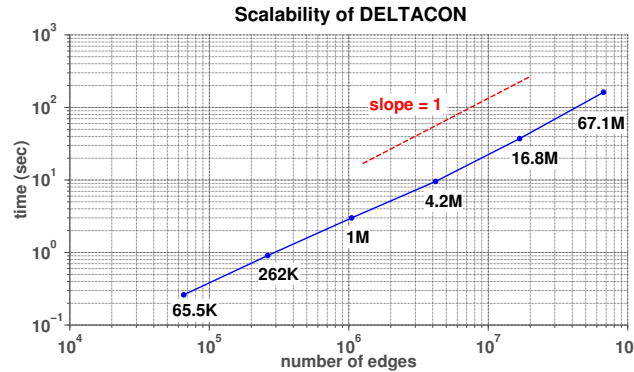


Fig. 7: DELTACON is linear on the number of edges (time in sec. vs. number of edges). The exact number of edges is annotated.

5.4. Robustness of DELTACON

DELTACON₀ satisfies all the desired properties, but its runtime is quadratic and does not scale well to large graphs with more than a few million edges. On the other hand, our second proposed algorithm, DELTACON, is scalable both in theory and practice (lemma 2, section 5.3). In this section we present the sensitivity of DELTACON to the number of groups g , as well as how the similarity scores of DELTACON and DELTACON₀ compare.

For this experiment, we use complete and star graphs, as well as the Political Blogs dataset. For each of the synthetic graphs (complete graph with 100 nodes and star graph with 100 nodes), we create three corrupted versions, where we remove 1, 3 and 10 edges respectively. For the real dataset, we create four corrupted versions of the Political Blogs graph by removing {10%, 20%, 40%, 80%} of the edges. For each pair of <original, corrupted> graphs, we compute the DELTACON similarity for varying number of groups. We note that when $g = n$, DELTACON is equivalent

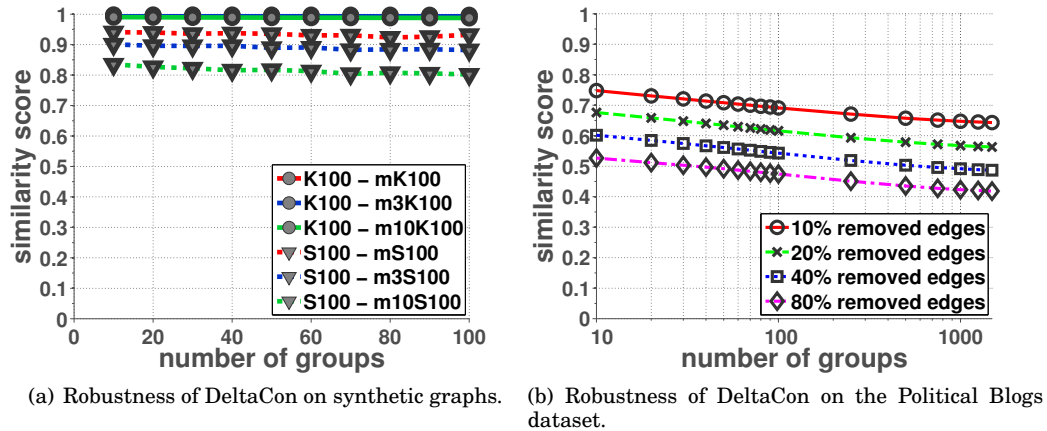


Fig. 8: DELTACON is robust to the number of groups. And more importantly, at every group level, the ordering of similarities of the different graph pairs remains the same (e.g., $\text{sim}(K100, m1K100) > \text{sim}(K100, m3K100) > \dots > \text{sim}(S100, m1S100) > \dots > \text{sim}(S100, m10S100)$).

to DELTACON₀. The results for the synthetic and real graphs are shown in Figures 8(a) and 8(b) respectively.

OBSERVATION 8. In our experiments, DELTACON₀ and DELTACON agree on the ordering of pairwise graph similarities.

In Fig. 8(b) the lines not only do not cross, but are almost parallel to each other. This means that for a fixed number of groups g , the differences between the similarities of different graph pairs remain the same. In other words, the ordering of similarities is the same at every group level g .

OBSERVATION 9. The similarity scores of DELTACON are robust to the number of groups, g .

Obviously the bigger the number of groups, the closer are the similarity scores to the “ideal” similarity scores, i.e. scores of DELTACON₀. For instance, in Fig. 8(b), when each blog is in its own group ($g = n = 1490$), the similarity scores between the original network and the derived networks (with any level of corruption) are identical to the scores of DELTACON₀. However, even with few groups, the approximation of DELTACON₀ is very good.

It is worth mentioning that the bigger the number of groups g , the bigger runtime is required, since the complexity of the algorithm is $O(g \cdot \max\{m1, m2\})$. Not only the accuracy, but also the runtime increases with the number of groups; so, the speed and accuracy trade-offs need to be conciliated. Experimentally, it seems that a good compromise is achieved even for g smaller than 100.

6. DELTACON & DELTACON-ATTR AT WORK

In this section we present three applications of our graph similarity algorithms, one of which comes from social networks and the other two from the area of neuroscience.

6.1. Enron.

Graph Similarity. First, we employ DELTACON to analyze the ENRON dataset, which consists of emails sent among employees in a span of more than two years. Figure 9 depicts the DELTACON similarity scores between consecutive daily who-emailed-whom graphs. By applying Quality Control with Individual Moving Range, we obtain the lower and upper limits of the

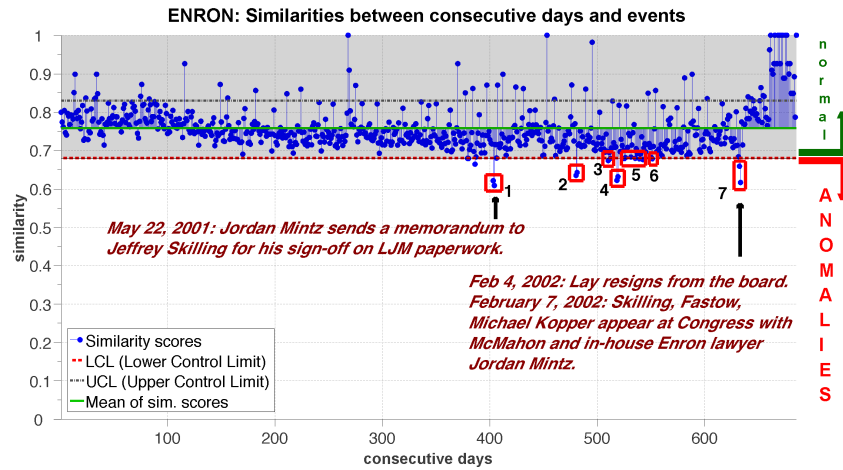


Fig. 9: DELTACON detects anomalies on Enron data coinciding with major events. The marked days correspond to anomalies. The blue points are similarity scores between consecutive instances of the daily email activity between the employees, and the marked days are 3σ units away from the median similarity score.

in-control similarity scores. These limits correspond to median $\pm 3\sigma$ ⁵. Using this method we were able to define the threshold (lower control limit) below which the corresponding days are anomalous, i.e. they differ “too much” from the previous and following days. Note that all the anomalous days relate to crucial events in the company’s history in 2001 (points marked with red boxes in Fig. 9):

- (1) May 22nd, 2001. Jordan Mintz sends a memorandum to Jeffrey Skilling (CEO for a few months) for his sign-off on LJM paperwork.
- (2) August 21st, 2001. Kenneth Lay, the CEO of Enron, emails all employees stating he wants “to restore investor confidence in Enron.”;
- (3) September 26th, 2001. Lay tells employees that the accounting practices are “legal and totally appropriate”, and that the stock is “an incredible bargain.”;
- (4) October 5th, 2001. Just before Arthur Andersen hired Davis Polk & Wardwell law firm to prepare a defense for the company;
- (5) October 24-25th, 2001. Jeff McMahon takes over as CFO. Email to all employees states that all the pertinent documents should be preserved;
- (6) November 8th, 2001. Enron announces it overstated profits by 586 million dollars over five years.
- (7) February 4th, 2002: Lay resigns from board.

Although high similarities between consecutive days do not consist anomalies, we found that mostly weekends expose high similarities. For instance, the first two points of 100% similarity correspond to the weekend before Christmas in 2000 and a weekend in July, when only two employees sent emails to each other. It is noticeable that after February 2002 many consecutive days are very similar; this happens because, after the collapse of Enron, the email exchange activity was rather low and often between certain employees.

Attribution. We additionally apply DELTACON-ATTR to the ENRON dataset for the months of May 2001 and February 2002, which are the most anomalous months according to the analysis of the data on a month-to-month timescale. Based on the node and edge rankings produced as a result of our method, we drew some interesting real-world conclusions .

⁵The median is used instead of the mean, since appropriate hypothesis tests demonstrate that the data does not follow the normal distribution. Moving range mean is used to estimate σ .

May 2001:

- Top Influential Culprit: John Lavorato, the former head of Enron's trading operations and CEO of Enron America, connected to ~50 new nodes in this month.
- Second Most Influential Culprit: Andy Zipper, VP of Enron Online, maintained contact with all those from the previous month, but also connected to 12 new people.
- Third Most Influential Culprit: Louise Kitchen, another employee (President of ENRON Online) lost 5-6 connections and made 5-6 connections. Most likely, some of the connections she lost or made were very significant in terms of expanding/reducing her office network.

February 2002:

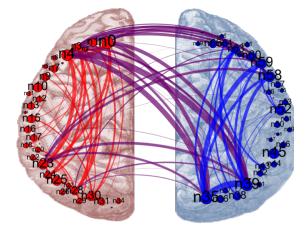
- Top Influential Culprit: Liz Taylor lost 51 connections this month but made no new ones - it is reasonable to assume that she likely quit the position or was fired (most influential culprit)
- Second Most Influential Culprit: Louise Kitchen (third culprit in May 2001) made no new connections, but lost 22 existing ones.
- Third Most Influential Culprit: Stan Horton (CEO of Enron Transportation) made 6 new connections and lost none. Some of these connections are likely significant in terms of expanding his office network.
- Fourth, Fifth and Sixth Most Influential Culprits: Employees Kam Keiser, Mike Grigsby (former VP for Enron's Energy Services) and Fletcher Sturm (VP) all lost many connections and made no new ones. Their situations were likely similar to those of Liz Taylor and Louise Kitchen.

6.2. Brain Connectivity Graph Clustering.

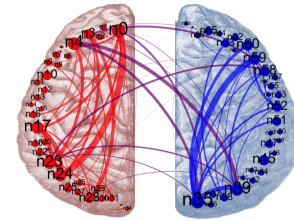
We also use DELTACON for clustering and classification of graphs. For this purpose we study *connectomes*, i.e., brain graphs, which are obtained by Multimodal Magnetic Resonance Imaging [Gray et al. 2012].

In total we study the connectomes of 114 people which are related to attributes such as age, gender, IQ. Each brain graph consists of 70 cortical regions (nodes), and connections (weighted edges) between them (see Table VI “Brain Graphs Small”). We ignore the strength of connections and derive an undirected, unweighted brain graph per person.

We first compute the DELTACON pairwise similarities between the brain graphs, and then perform hierarchical clustering using Ward's method (Fig. 1(b)). As shown in the figure, there are two clearly separable groups of brain graphs. Applying t-test on the available attributes for the two groups created by the clusters, we found that the latter differ significantly ($p < .01$) in the Composite Creativity Index (CCI), which is related to the person's performance on a series of creativity tasks. Figure 10 illustrates the brain connections in a subject with high and low creativity index. It appears that more creative subjects have more and heavier connections across the hemispheres than subjects that are less creative. Moreover, the two groups correspond to significantly different openness index ($p = .0558$), one of the “Big Five Factors”; that is, the brain connectivity is different in people that are inventive and people that are consistent. Exploiting analysis of variance (ANOVA: generalization of t-test when more than 2 groups are analyzed), we tested whether the various clusters that we obtain from the connectivity-based hierarchical map to differences in other attributes. However, in the dataset we studied there is no sufficient statistical evidence that age, gender, IQ etc. are related to the brain connectivity.



(a) Brain graph of subject with high creativity index.



(b) Brain graph of subject with low creativity index.

Fig. 10: Illustration of brain graphs for subjects with low and high creativity index resp. The low-CCI brain has fewer and lighter cross-hemisphere connections than the high-CCI brain.

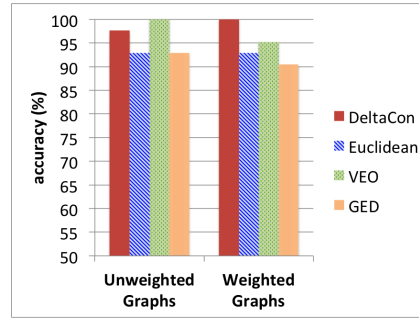


Fig. 11: DELTACON similarity outperforms almost all the baseline approaches. In the figure, the accuracy reflects the percentage of correctly recovered pairs of brain graphs. DELTACON recovers correctly *all* pairs of connectomes that correspond to the same subject for *weighted* graphs outperforming all the baselines. It also recovers almost all the pairs correctly in the case of *unweighted* graphs, following VEO which has the best accuracy. The variants of λ -distance perform very poorly (accuracy $\sim 2.38\%$) and are omitted from the plot, while the signature similarity method runs out of memory for the large input graphs.

6.3. Test-Retest: Big Brain Connectomes.

We also applied our method on the KKI-42 dataset [Roncal et al. 2013; OCP 2014], which consists of connectomes corresponding to $n = 21$ subjects. Each subject underwent two Functional MRI scans at different times, and so the dataset has $2n = 42$ large connectomes with ~ 17 million voxels and 49.3 to 90.4 million connections among them (see Table VI “Brain Graphs Big”). Our goal is to recover the pairs of connectomes that correspond to the same subject, by relying only on the structures of the brain graphs. In the following analysis, we compare our method to the standard approach in the neuroscience literature [Roncal et al. 2013], the Euclidean distance (as induced by the Frobenius norm), and also to the baseline approaches we introduced in Section 5.

We ran the following experiments on a 32-cores Intel(R) Xeon(R) CPU E7-8837 at 2.67GHz, with 1TB of RAM. The signature similarity method runs out of memory for these large graphs, and we could not use it for this application. Moreover, the variants of λ -distance are computationally expensive, even when we compute only a few top eigenvalues, and they also perform very poorly in this task.

Unweighted graphs. The brain graphs that are obtained from the FMRI scans have weighted edges that correspond to the strength of the connections between different voxels. The weights tend to be noisy, so we initially ignore them, and treat the brain graphs as binary by mapping all the non-zero weights to 1. To discover the pairs of connectomes that correspond to the same subject, we first find the DELTACON pairwise similarities between the connectomes. We note that it suffices to do $\binom{2n}{2} = 861$ graph comparisons, since the DELTACON similarities are symmetric. Then, we find the potential pairs of connectomes that belong to the same subject by using the following approach: For each connectome $C_i \in \{1, \dots, 2n\}$, we choose the connectome $C_j \in \{1, \dots, 2n\} \setminus i$ s.t. $\text{sim}(C_i, C_j)$ is maximized. In other words, we pair each connectome with its most similar graph (excluding itself) as defined by DELTACON. This results in 97.62% accuracy of predicting the connectomes of the same subject.

In addition to our method, we compute the pairwise *Euclidean distances* (ED) between the connectomes and evaluate the predictive power of ED. Specifically, we compute the quantities $ED(i, j) = \|C_i - C_j\|_F^2$, where C_i and C_j are the binary adjacency matrices of the connectomes i and j respectively, and $\|\cdot\|_F$ is the Frobenius norm of the enclosed matrix. As before, for each connectome $i \in \{1, \dots, 2n\}$, we choose the connectome $j \in \{1, \dots, 2n\} \setminus i$ s.t. $ED(i, j)$ is minimized⁶,

⁶We note that DELTACON computes the similarity between two graphs, while ED computes their distance. Thus, when trying to find the pairs of connectomes that belong to the same subject, we want to maximize the similarity, or equivalently, minimize the distance.

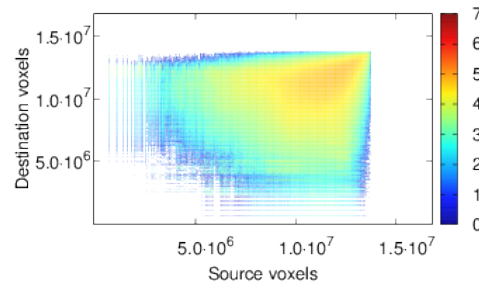
the accuracy of recovering the pairs of connectomes that correspond to the same subject is 92.86% (vs. 97.62% for DELTACON) as shown in Fig. 11.

Finally, from the baseline approaches, the Vertex/Edge Overlap similarity performs slightly better than our method, while GED has the same accuracy as the Euclidean distance (Fig. 11, 'Unweighted Graphs'). All the variants of λ -distance perform very poorly with 2.38% accuracy. As mentioned before, the signature similarity runs out of memory and, hence, could not be used for this application.

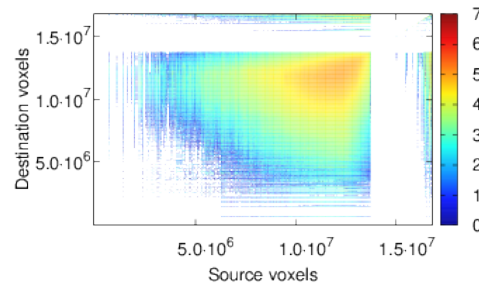
Weighted graphs. We also wanted to see how accurately the methods can recover the pairs of *weighted* connectomes that belong to the same subject. Given that the weights are noisy, we follow the common practice and first smooth them by applying the logarithm function (with base 10). Then we follow the procedure described above both for DELTACON and the Euclidean distance. Our method yields 100% accuracy in recovering the pairs, while the Euclidean distance results in 92.86% accuracy. The results are shown in Fig. 11 (labeled 'Weighted Graphs'), while Fig. 12 shows a case of brain graphs that were incorrectly recovered by the ED-based method, but successfully found by DELTACON.

In the case of weighted graphs, as shown in Fig. 11, all the baseline approaches perform worse than DELTACON in recovering the correct brain graph pairs. In the plot we include the methods that have comparable performance to our method. The λ -distance has the same very poor accuracy (2.38% for all the variants) as in the case of unweighted graphs, while the signature similarity could not be applied due to its very high memory requirements.

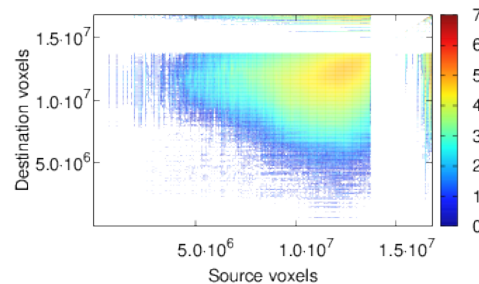
Therefore, by using DELTACON we are able to recover, with almost perfect accuracy, which large connectomes belong to the same subject. On the other hand, the commonly used technique, the Euclidean distance, as well as the baseline methods fail to detect several connectome pairs (with the exception of VEO in the case of unweighted graphs).



(a) Test brain graph of 32 year old male.



(b) True re-test brain graph of the same male as in (a).



(c) Recovered re-test brain graph by the Euclidean distance.

Fig. 12: DELTACON outperforms the Euclidean distance in recovering the correct test-retest pairs of brain graphs. We depict the spy plots of three brain graphs, across which the order of nodes is the same and corresponds to increasing order of degrees for the leftmost spy plot. (a) and (b) are the test and re-test graphs of a male who is 32 years old, while (c) is the re-test brain graph of a 38 years old male. The correct test-retest pair (a)-(b) that DELTACON recovers consists of “visually” more similar brain graphs than the incorrect test-retest pair (a)-(c) that Euclidean distance found.

7. RELATED WORK

The related work comprises three main areas: Graph similarity, node affinity algorithms, and temporal anomaly detection with node attribution. We give the related work in each area separately, and mention what sets our method apart.

Graph Similarity. Graph similarity refers to the problem of quantifying how similar are two graphs. The graphs may have the same or different sets of nodes, and can be divided in two main categories:

(1) *With Known Node Correspondence.* The first category assumes that the two given graphs are aligned, or, in other words, the node correspondence between the two graphs is given. [Papadimitriou et al. 2008] proposes five similarity measures for directed web graphs, which are applied for anomaly detection. Among them the best is the Signature Similarity (SS), which is based on the SimHash algorithm, while the Vertex/Edge Overlap similarity (VEO) performs very well. Bunke [Bunke et al. 2006] presents techniques used to track sudden changes in communications networks for performance monitoring. The best approaches are the Graph Edit Distance and Maximum Common Subgraph. Both are NP-complete, but the former approach can be simplified given the application and it becomes linear on the number of nodes and edges in the graphs. The current manuscript attacks the graph similarity problem with known node correspondence, and is an extension of the work in [Koutra et al. 2013b], where DELTACON was first introduced. In addition to computational methods to assess the similarity between graphs, there is also a line of work on visualization-based graph comparison. These techniques are based on side-by-side visualization of the two networks [Andrews et al. 2009; Hascoët and Dragicevic 2012], or superimposed/augmented graph or matrix views [Alper et al. 2013; Erten et al. 2004]. A review of visualization-based comparison of information based on these and additional techniques is given in [Gleicher et al. 2011]. [Alper et al. 2013] investigate ways of visualizing differences between small brain graphs using either augmented graph representations or augmented adjacency matrices. However, their approach works for small and sparse graphs (40-80 nodes). Honeycomb [van Ham et al. 2009] is a matrix-based visualization tool that handles larger graphs with several thousands edges, and performs temporal analysis of a graph by showing the time series of graph properties that are of interest. The visualization methods do not compute the similarity score between two graphs, but show just their differences. This is related to the culprit nodes and edges that our method DELTACON finds. However, these methods tend to visualize all the differences between two graphs, while our algorithm routes attention to the nodes and edges that are mostly responsible for the differences among the input graphs. All in all, visualizing and comparing graphs with millions or billions of nodes and edges remains a challenge, and best suits small problems.

(2) *With Unknown Node Correspondence.* The previous works assume that the correspondence of nodes across the two graphs is known, but this is not always the case. Social network analysis, bioinformatics, and pattern recognition are just a few domains with applications where the node correspondence information is missing or even constitutes the objective. The works attacking this problem can be divided in three main approaches: (a) feature extraction and similarity computation based on the feature space, (b) graph matching and application of techniques from the first category, and (c) graph kernels.

There are numerous works that fall follow the first approach and use features to define the similarity between graphs. The λ -distance, a spectral method which defines the distance between two graphs as the distance between their spectra (eigenvalues) has been studied thoroughly ([Bunke et al. 2006; Peabody 2003; Wilson and Zhu 2008], algebraic connectivity [Fiedler 1973]). The existence of co-spectral graphs with different structure, and the big differences in the graph spectra, despite subtle changes in the graphs, are two weaknesses that add up to the high computational cost of computing the whole graph spectrum. Clearly, the spectral methods that call for the whole spectrum cannot scale to the large-scale graphs with billions of nodes and edges that are of interest currently. Also, depending on the graph-related matrix that is considered (adjacency, laplacian, normalized laplacian), the distance between the graphs is different. As we show in Section 5, these methods fail to satisfy one or more of the desired properties for graph comparison. [Li et al. 2011] proposes an SVM-based approach on some global feature vectors (including the average degree, eccentricity, number of nodes and edges, num-

ber of eigenvalues, and more) of the graphs in order to perform graph classification. Macindoe and Richards [Macindoe and Richards 2010] focus on social networks and extract three socially relevant features: the leadership, bonding and diversity. The complexity of the latter feature makes the method applicable to graphs with up to a few million edges. Other techniques include computing edge curvatures under heat kernel embedding [Elghawalby and Hancock 2008], comparing the number of spanning trees [Kelmans 1976], comparing graph signatures consisting of summarized local structural features [Berlingerio et al. 2013], and a distance based on graphlet correlation [Yaveroğlu et al. 2014].

The second approach first solves the graph matching or alignment problem – i.e., finds the ‘best’ correspondence between the nodes of the two graphs– and then finds the distance (or similarity) between the graphs. [Conte et al. 2004] reviews graph matching algorithms in pattern recognition. There are over 150 publications that attempt to solve the graph alignment problem under different settings and constraints. The methods span from genetic algorithms to decision trees, clustering, expectation-maximization and more. Some recent methods that are more efficient for large graphs include a distributed, belief-propagation based method for protein alignment [Bradde et al. 2010], another message-passing algorithm for aligning sparse networks when some possible matchings are given [Bayati et al. 2013], and a gradient-descent based method for aligning probabilistically large bipartite graphs [Koutra et al. 2013a].

The third approach uses kernels between graphs, which were introduced in 2010 by [Vishwanathan et al. 2010]. Graph kernels work directly on the graphs without doing feature extraction. They compare graph structures, such as walks [Kashima et al. 2003; Gärtner et al. 2003], paths [Borgwardt and Kriegel 2005], cycles [Horváth et al. 2004], trees [Ramon and Grtner 2003; Mahé and Vert 2009], graphlets [Shervashidze et al. 2009; Costa and De Grave 2010] which can be computed in polynomial time. A popular graph kernel is the random walk graph kernel [Kashima et al. 2003; Gärtner et al. 2003], which finds the number of common walks on the two input graphs. The simple version of this kernel is slow, requiring $O(n^6)$ runtime, but can be sped up to $O(n^3)$ by using the Sylvester equation. In general, the above-mentioned graph kernels do not scale well to graphs with more than 100 nodes. A faster implementation of the random walk graph kernel with $O(n^2)$ runtime was proposed by Kang et al. [Kang et al. 2012]. The fastest kernel to date is the subtree kernel proposed by Shervashidze and Borgwardt [Shervashidze and Borgwardt 2009; Shervashidze et al. 2011], which is linear on the number of edges and the maximum degree, $O(m \cdot d)$, in the graphs. The proposed approach uses the Weisfeiler-Lehman test of isomorphism, and operates on *labeled* graphs. In our work, we consider large, unlabeled graphs, while most kernels require at least $O(n^3)$ runtime or labels on the nodes/edges. Thus, we do not compare them to DELTACON quantitatively.

Both research problems – graph similarity with given or missing node correspondence– are important, but apply in different settings. If the node correspondence is available, the algorithms that make use of it can only perform better than methods that omit it. Our work tackles the former problem.

Node Affinity. There are numerous node affinity algorithms; Pagerank [Brin and Page 1998], Personalized Random Walks with Restarts [Haveliwala 2003], the electric network analogy [Doyle and Snell 1984], SimRank [Jeh and Widom 2002] and extensions/improvements [Yu et al. 2013], [Li et al. 2010], and Belief Propagation [Yedidia et al. 2003] are only some examples of the most successful techniques. Here we focus on the latter method, and specifically a fast variation [Koutra et al. 2011] which is also intuitive. All the techniques have been used successfully in many tasks, such as ranking, classification, malware and fraud detection ([Chau et al. 2011], [McGlohon et al. 2009]), and recommendation systems [Kim and El Saddik 2011].

Anomaly Detection. Anomaly detection in static graphs has been studied using various data mining and statistical techniques [Akoglu et al. 2014; Khoa and Chawla 2010; Akoglu et al. 2012; Lee et al. 2013; Kang et al. 2014]. Detection of anomalous behaviors in time-evolving networks is more relevant to our work, and is covered in the surveys [Akoglu et al. 2014; Ranshous et al. 2015]. A non-inclusive list of works on temporal graph anomaly detection follows. [Maruhashi et al. 2011], [Koutra et al. 2012] and [Mao et al. 2014] employ tensor decomposition to identify anomalous substructures in graph data in the context of intrusion detection. Henderson et al. propose a multi-level approach for identifying anomalous behaviors

in volatile temporal graphs based on iteratively pruning the temporal space using multiple graph metrics [Henderson et al. 2010]. CopyCatch [Beutel et al. 2013] is a clustering-based MapReduce approach to identify lockstep behavior in Page Like patterns on Facebook. Akoglu and Faloutsos use local features and the node eigen-behaviors to detect points of change – when many of the node behave differently –, and also spot nodes that are mostly responsible for the change point [Akoglu and Faloutsos 2010]. Finally, [Sricharan and Das 2014] monitors changes in the commute time between all pairs of nodes to detect anomalous nodes and edges in time-evolving networks. All these works use various approaches to detect anomalous behaviors in dynamic graphs, though they are not based on the similarity between graphs, which is the focus of our work.

Node/Edge Attribution. Some of the anomaly detection methods discover anomalous nodes, and other anomalous structures in the graphs. In a slightly different context, a number of techniques have been proposed in the context of node and edge importance in graphs. PageRank, HITS [Kleinberg 1999] and betweenness centrality (random-walk based [Newman 2005] and shortest-path based [Freeman 1977]) are several such methods for the purpose of identifying important nodes. [Tong et al. 2012] proposes a method to determine edge importance for the purpose of augmenting or inhibiting dissemination of information between nodes. To the best of our knowledge, this and other existing methods focus only on identifying important nodes and edges in the context of a single graph. In the context of anomaly detection, [Akoglu and Faloutsos 2010] and [Sricharan and Das 2014] detect nodes that contribute mostly to change events in time-evolving networks.

Among these works, the most relevant to ours are the methods proposed by [Akoglu and Faloutsos 2010] and [Sricharan and Das 2014]. The former relies on the selection of features, and tends to return a large number of false positives. Moreover, because of the focus on local egonet features, it may not distinguish between small and large changes in time-evolving networks [Sricharan and Das 2014]. At the same time, and independently from us, Sricharan and Das proposed CAD [Sricharan and Das 2014], a method which defines the anomalousness of edges based on the commute time between nodes. The commute time is the expected number of steps in a random walk starting at i , before node j is visited and then node i is reached again. This method is closely related to DELTACON as Belief Propagation (the heart of our method), and Random Walks with Restarts (the heart of CAD) are equivalent under certain conditions [Koutra et al. 2011]. However, the methods work in different directions: DELTACON first identifies the most anomalous nodes, and defines the anomalousness of edges as a function of the outlieriness of the adjacent nodes; CAD first identifies the most anomalous edges, and then defines as anomalous all their adjacent nodes without ranking them. Our method does not only find anomalous nodes and edges in a graph, but also (i) ranks them in decreasing order of anomalousness (which can be used for guiding attention to important changes) and (ii) quantifies the difference between two graphs (which can also be used for graph classification, clustering and other tasks).

8. CONCLUSIONS

In this work, we tackle the problem of graph similarity when the node correspondence is known, such as similarity in time-evolving phone networks. Our contributions are:

- *Axioms/Properties:* we formalize the problem of graph similarity by providing axioms, and desired properties.
- *Algorithm:* We propose DELTACON, a graph similarity algorithm that is (a) *principled* (axioms A1-A3, in Sec. 2), (b) *intuitive* (properties P1-P4, in Sec. 5), and (c) *scalable*, needing on commodity hardware ~160 seconds for a graph with over 67 million edges. We additionally introduce DELTACON-ATTR, a scalable method enabling node/edge attribution for differences between graphs.
- *Experiments:* We evaluate the intuitiveness of DELTACON, and compare it to 6 state-of-the-art measures by using various synthetic and real, big graphs.
- *Applications:* We use DELTACON and DELTACON-ATTR for temporal anomaly detection (ENRON), clustering & classification (brain graphs), as well as recovery of test-retest brain scans.

Future directions include extending DELTACON to handle streaming graphs, incremental similarity updates for time-evolving graphs, as well as graphs with node and/or edge attributes. Revisiting the desired properties, adapting them to these types of graphs, and exploring the trade-off between approximate computation of the similarity score and the runtime are only a few of the points worth investigating.

ACKNOWLEDGMENTS

The authors would like to thank William Cohen, Markos Koutras, Elijah Mayfield, Gary Miller, Dana Movshovitz-Attias, Aaditya Ramdas, Aarti Singh, and Jilles Vreeken for their helpful comments and suggestions.

Funding was provided by the U.S. Army Research Office (ARO) and Defense Advanced Research Projects Agency (DARPA) under Contract Number W911NF-11-C-0088. This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344. Research was also sponsored by the Army Research Laboratory and was accomplished under Cooperative Agreement Number W911NF-09-2-0053. It was also partially supported by an IBM Faculty Award, a Google Focused Research Award, and a Yahoo Research Alliance Gift. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

REFERENCES

- Leman Akoglu, Duen Horng Chau, U Kang, Danai Koutra, and Christos Faloutsos. 2012. Opavion: Mining and visualization in large graphs. In *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*. ACM, 717–720.
- Leman Akoglu and Christos Faloutsos. 2010. Event Detection in Time Series of Mobile Communication Graphs. In *27th Army Science Conference*.
- Leman Akoglu, Hanghang Tong, and Danai Koutra. 2014. Graph-based Anomaly Detection and Description: A Survey. *Data Mining and Knowledge Discovery (DAMI)* (April 2014).
- David Aldous and James Allen Fill. 2002. Reversible Markov Chains and Random Walks on Graphs. (2002). Unfinished monograph, recompiled 2014, available at [http://www.stat.berkeley.edu/~sim\\$aldous/RWG/book.html](http://www.stat.berkeley.edu/~sim$aldous/RWG/book.html).
- Basak Alper, Benjamin Bach, Nathalie Henry Riche, Tobias Isenberg, and Jean-Daniel Fekete. 2013. Weighted Graph Comparison Techniques for Brain Connectivity Analysis. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '13)*. ACM, New York, NY, USA, 483–492.
- K. Andrews, M. Wohlfahrt, and G. Wurzinger. 2009. Visual Graph Comparison. In *13th International Conference in Information Visualization*. 62–67.
- Mohsen Bayati, David F. Gleich, Amin Saberi, and Ying Wang. 2013. Message-Passing Algorithms for Sparse Network Alignment. *ACM Trans. Knowl. Discov. Data* 7, 1 (March 2013), 3:1–3:31.
- Michele Berlingerio, Danai Koutra, Tina Eliassi-Rad, and Christos Faloutsos. 2013. Network similarity via multiple social theories. In *Proceedings of the 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*. ACM, 1439–1440.
- Alex Beutel, Wanhong Xu, Venkatesan Guruswami, Christopher Palow, and Christos Faloutsos. 2013. Copy-Catch: stopping group attacks by spotting lockstep behavior in social networks. In *WWW*. ACM, 119–130.
- Karsten M. Borgwardt and Hans-Peter Kriegel. 2005. Shortest-Path Kernels on Graphs. In *Proceedings of the Fifth IEEE International Conference on Data Mining (ICDM '05)*. IEEE Computer Society, Washington, DC, USA, 74–81.
- S. Bradde, A. Braunstein, H. Mahmoudi, F. Tria, M. Weigt, and R. Zecchina. 2010. Aligning graphs and finding substructures by a cavity approach. *Europhysics Letters* 89 (2010). Issue 3.
- Sergey Brin and Lawrence Page. 1998. The Anatomy of a Large-Scale Hypertextual Web Search Engine. *Computer Networks* 30, 1-7 (1998), 107–117.
- Horst Bunke, Peter J. Dickinson, Miro Kraetzl, and Walter D. Wallis. 2006. *A Graph-Theoretic Approach to Enterprise Network Dynamics (PCS)*. Birkhauser.
- Rajmonda Sulo Caceres, Tanya Y. Berger-Wolf, and Robert Grossman. 2011. Temporal Scale of Processes in Dynamic Networks. In *Proceedings of the International Workshop on Data Mining in Networks at the IEEE International Conference on Data Mining (ICDM), Vancouver*. 925–932.

- Duen Horng Chau, Carey Nachenberg, Jeffrey Wilhelm, Adam Wright, and Christos Faloutsos. 2011. Large Scale Graph Mining and Inference for Malware Detection. In *Proceedings of the 11th SIAM International Conference on Data Mining (SDM)*, Mesa, AZ. 131–142.
- Yihua Chen, Eric K. Garcia, Maya R. Gupta, Ali Rahimi, and Luca Cazzanti. 2009. Similarity-based Classification: Concepts and Algorithms. *J. Mach. Learn. Res.* 10 (June 2009), 747–776.
- D. Conte, P. Foggia, C. Sansone, and M. Vento. 2004. Thirty Years Of Graph Matching In Pattern Recognition. *International Journal of Pattern Recognition and Artificial Intelligence* (2004).
- Fabrizio Costa and Kurt De Grave. 2010. Fast neighborhood subgraph pairwise distance kernel. In *Proceedings of the 26th International Conference on Machine Learning*.
- Peter G. Doyle and J. Laurie Snell. 1984. *Random Walks and Electric Networks*. Mathematical Association of America.
- Hewayda Elghawalby and Edwin R. Hancock. 2008. Measuring Graph Similarity Using Spectral Geometry. In *Proceedings of the 5th international conference on Image Analysis and Recognition (ICIAR)*. 517–526.
- Cesim Erten, PhilipJ. Harding, StephenG. Kobourov, Kevin Wampler, and Gary Yee. 2004. GraphAEL: Graph Animations with Evolving Layouts. In *Graph Drawing*. Lecture Notes in Computer Science, Vol. 2912. Springer Berlin Heidelberg, 98–110.
- M. Fiedler. 1973. Algebraic connectivity of graphs. *Czechoslovak Mathematical Journal* 23, 98 (1973), 298–305.
- Linton C Freeman. 1977. A set of measures of centrality based on betweenness. *Sociometry* (1977), 35–41.
- Keinosuke Fukunaga. 1990. *Introduction to statistical pattern recognition*. Access Online via Elsevier.
- Thomas Gärtner, Peter A. Flach, and Stefan Wrobel. 2003. On Graph Kernels: Hardness Results and Efficient Alternatives. In *Proceedings of the 16th Annual Conference on Computational Learning Theory and the 7th Kernel Workshop*.
- Michael Gleicher, Danielle Albers Szafr, Rick Walker, Ilir Jusufi, Charles D. Hansen, and Jonathan C. Roberts. 2011. Visual Comparison for Information Visualization. *Information Visualization* 10, 4 (Oct 2011), 289–309.
- W.R. Gray, J.A. Bogovic, J.T. Vogelstein, B.A. Landman, J.L. Prince, and R.J. Vogelstein. 2012. Magnetic Resonance Connectome Automated Pipeline: An Overview. *IEEE Pulse* 3, 2 (2012), 42–48.
- R. Guha, Ravi Kumar, Prabhakar Raghavan, and Andrew Tomkins. 2004. Propagation of trust and distrust. In *Proceedings of the 13th International Conference on World Wide Web (WWW)*, New York, NY. ACM, 403–412.
- Mountaz Hascoët and Pierre Dragicevic. 2012. Interactive Graph Matching and Visual Comparison of Graphs and Clustered Graphs. In *Proceedings of the International Working Conference on Advanced Visual Interfaces (AVI '12)*. ACM, New York, NY, USA, 522–529.
- Taher H. Haveliwala. 2003. Topic-Sensitive PageRank: A Context-Sensitive Ranking Algorithm for Web Search. *IEEE Transactions on Knowledge and Data Engineering* 15, 4 (2003), 784–796.
- Keith Henderson, Tina Eliassi-Rad, Christos Faloutsos, Leman Akoglu, Lei Li, Koji Maruhashi, B. Aditya Prakash, and Hanghang Tong. 2010. Metric forensics: a multi-level approach for mining volatile graphs.. In *KDD*, Bharat Rao, Balaji Krishnapuram, Andrew Tomkins, and Qiang Yang 0001 (Eds.). ACM, 163–172.
- Tamás Horváth, Thomas Gärtner, and Stefan Wrobel. 2004. Cyclic Pattern Kernels for Predictive Graph Mining. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '04)*. ACM, New York, NY, USA, 158–167.
- Glen Jeh and Jennifer Widom. 2002. SimRank: a measure of structural-context similarity. In *Proceedings of the 8th ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, Edmonton, Alberta. ACM, 538–543.
- U. Kang, Jay Yoon Lee, Danai Koutra, and Christos Faloutsos. 2014. Net-Ray: Visualizing and Mining Billion-Scale Graphs. In *Advances in Knowledge Discovery and Data Mining - 18th Pacific-Asia Conference, PAKDD 2014, Tainan, Taiwan, May 13-16, 2014. Proceedings, Part I*. 348–361.
- U. Kang, Hanghang Tong, and Jimeng Sun. 2012. Fast Random Walk Graph Kernel. In *Proceedings of the 12th SIAM International Conference on Data Mining (SDM)*.
- George Karypis and Vipin Kumar. 1995. Metis-unstructured graph partitioning and sparse matrix ordering system, version 2.0. (1995).
- Hisashi Kashima, Koji Tsuda, and Akihiro Inokuchi. 2003. Marginalized kernels between labeled graphs. In *Proceedings of the Twentieth International Conference on Machine Learning*. AAAI Press, 321–328.
- A.K. Kelmans. 1976. Comparison of graphs by their number of spanning trees. *Discrete Mathematics* 16, 3 (1976), 241 – 261.

- Nguyen Lu Dang Khoa and Sanjay Chawla. 2010. Robust Outlier Detection Using Commute Time and Eigenspace Embedding.. In *PAKDD (2)* (2010-06-08) (*Lecture Notes in Computer Science*), Mohammed Javeed Zaki, Jeffrey Xu Yu, B. Ravindran, and Vikram Pudi (Eds.), Vol. 6119. Springer, 422–434.
- Heung-Nam Kim and Abdulmotaleb El Saddik. 2011. Personalized PageRank vectors for tag recommendations: inside FolkRank. In *Proceedings of the fifth ACM conference on Recommender systems*. 45–52.
- Jon M Kleinberg. 1999. Authoritative sources in a hyperlinked environment. *Journal of the ACM (JACM)* 46, 5 (1999), 604–632.
- Bryan Klimt and Yiming Yang. 2004. Introducing the Enron Corpus.
- Danai Koutra, Tai-You Ke, U. Kang, Duen Horng Chau, Hsing-Kuo Kenneth Pao, and Christos Faloutsos. 2011. Unifying Guilt-by-Association Approaches: Theorems and Fast Algorithms. In *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD), Athens, Greece*.
- Danai Koutra, Evangelos E Papalexakis, and Christos Faloutsos. 2012. TensorSplat: Spotting Latent Anomalies in Time. In *Informatics (PCI), 2012 16th Panhellenic Conference on*. IEEE, 144–149.
- Danai Koutra, Hanghang Tong, and David Lubensky. 2013a. Big-Align: Fast Bipartite Graph Alignment.
- Danai Koutra, Joshua Vogelstein, and Christos Faloutsos. 2013b. DeltaCon: A Principled Massive-Graph Similarity Function.
- Jay Yoon Lee, U Kang, Danai Koutra, and Christos Faloutsos. 2013. Fast anomaly detection despite the duplicates. In *Proceedings of the 22nd international conference on World Wide Web companion*. International World Wide Web Conferences Steering Committee, 195–196.
- Jure Leskovec. 2013. Stanford Large Network Dataset Collection. <http://snap.stanford.edu/data/>. (2013).
- Jure Leskovec, Deepayan Chakrabarti, Jon M. Kleinberg, and Christos Faloutsos. 2005. Realistic, Mathematically Tractable Graph Generation and Evolution, Using Kronecker Multiplication. In *Proceedings of the 9th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD), Porto, Portugal*. 133–145.
- Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. 2007. Graph evolution: Densification and shrinking diameters. *IEEE Transactions on Knowledge and Data Engineering* 1 (March 2007).
- Cuiping Li, Jiawei Han, Guoming He, Xin Jin, Yizhou Sun, Yintao Yu, and Tianyi Wu. 2010. Fast computation of SimRank for static and dynamic information networks. In *Proceedings of the 13th International Conference on Extending Database Technology (EDBT '10)*. ACM, New York, NY, USA, 465–476.
- Geng Li, Murat Semerci, Bulent Yener, and M.J. Zaki. 2011. Graph Classification via Topological and Label Attributes. In *Proceedings of the 9th International Workshop on Mining and Learning with Graphs (MLG), San Diego, USA*.
- Owen Macindoe and Whitman Richards. 2010. Graph Comparison Using Fine Structure Analysis. In *Social-Com / PASSAT*. 193–200.
- Pierre Mahé and Jean-Philippe Vert. 2009. Graph Kernels Based on Tree Patterns for Molecules. *Mach. Learn.* 75, 1 (April 2009), 3–35.
- Ching-Hao Mao, Chung-Jung Wu, Evangelos E Papalexakis, Christos Faloutsos, and Tien-Cheu Kao. 2014. MalSpot: Multi² Malicious Network Behavior Patterns Analysis. In *PAKDD*.
- Koji Maruhashi, Fan Guo, and Christos Faloutsos. 2011. Multiaspectforensics: Pattern mining on large-scale heterogeneous networks with tensor analysis. In *ASONAM*. IEEE, 203–210.
- Mary McGlohon, Stephen Bay, Markus G. Anderle, David M. Steier, and Christos Faloutsos. 2009. SNARE: a link analytic system for graph labeling and risk detection. In *Proceedings of the 15th ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD), Paris, France*. 1265–1274.
- Mark EJ Newman. 2005. A measure of betweenness centrality based on random walks. *Social networks* 27, 1 (2005), 39–54.
- Caleb C. Noble and Diane J. Cook. 2003. Graph-based anomaly detection. In *Proceedings of the 9th ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD), Washington, DC*. 631–636.
- OCP. 2014. Open Connectome Project. <http://www.openconnectomeproject.org>. (2014).
- Panagiotis Papadimitriou, Ali Dasdan, and Hector Garcia-Molina. 2008. Web graph similarity for anomaly detection. *Journal of Internet Services and Applications* 1, 1 (2008), 1167.
- Mitchell Peabody. 2003. *Finding groups of graphs in databases*. Master’s thesis. Drexel University.
- Jan Ramon and Thomas Grtner. 2003. Expressivity versus efficiency of graph kernels. In *Proceedings of the First International Workshop on Mining Graphs, Trees and Sequences*. 65–74.

- Stephen Ranshous, Shitian Shen, Danai Koutra, Steven Harenberg, Christos Faloutsos, and Nagiza F. Samatova. 2015. Graph-based Anomaly Detection and Description: A Survey. *WIREs Computational Statistics* (January (accepted) 2015).
- William Gray Roncal, Zachary H. Koterba, Disa Mhembere, Dean Kleissas, Joshua T. Vogelstein, Randal C. Burns, Anita R. Bowles, Dimitrios K. Donavos, Sephira Ryman, Rex E. Jung, Lei Wu, Vince D. Calhoun, and R. Jacob Vogelstein. 2013. MIGRAINE: MRI Graph Reliability Analysis and Inference for Connectomics. *IEEE Global Conference on Signal and Information Processing (GlobalSIP)* (2013).
- Nino Shervashidze and Karsten Borgwardt. 2009. Fast Subtree Kernels on Graphs. In *Advances in Neural Information Processing Systems 22 – Proceedings of the 2001 Neural Information Processing Systems Conference NIPS 2009, December 7-10, 2009 Vancouver, British Columbia, Canada*, Yoshua Bengio, Dale Schuurmans, John Lafferty, Chris Williams, and Aron Culotta (Eds.). Neural Information Processing Systems Foundation, 1660–1668.
- Nino Shervashidze, Pascal Schweitzer, Erik Jan van Leeuwen, Kurt Mehlhorn, and Karsten M. Borgwardt. 2011. Weisfeiler-Lehman Graph Kernels. *J. Mach. Learn. Res.* 12 (Nov. 2011), 2539–2561.
- N. Shervashidze, S. V. N. Vishwanathan, T. Petri, K. Mehlhorn, and K. Borgwardt. 2009. Efficient graphlet kernels for large graph comparison. In *Proceedings of the Twelfth International Conference on Artificial Intelligence and Statistics (AISTATS 2009)*, David van Dyk and Max Welling (Eds.), Vol. 5. JMLR: W&CP, 488–495.
- Kumar Sricharan and Kamalika Das. 2014. Localizing Anomalous Changes in Time-evolving Graphs. In *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data (SIGMOD '14)*. ACM, New York, NY, USA, 1347–1358.
- Hanghang Tong, B Aditya Prakash, Tina Eliassi-Rad, Michalis Faloutsos, and Christos Faloutsos. 2012. Gelling, and melting, large graphs by edge manipulation. In *Proceedings of the 21st ACM international conference on Information and knowledge management*. ACM, 245–254.
- Frank van Ham, Hans-Jrg Schulz, and JoanM. Dimicco. 2009. Honeycomb: Visual Analysis of Large Scale Social Networks. In *Human-Computer Interaction INTERACT 2009*, Tom Gross, Jan Gulliksen, Paula Kotz, Lars Oestreicher, Philippe Palanque, RaquelOliveira Prates, and Marco Winckler (Eds.). Lecture Notes in Computer Science, Vol. 5727. Springer Berlin Heidelberg, 429–442.
- S. V. N. Vishwanathan, Nicol N. Schraudolph, Risi Imre Kondor, and Karsten M. Borgwardt. 2010. Graph Kernels. *Journal of Machine Learning Research* 11 (2010), 1201–1242.
- Bimal Viswanath, Alan Mislove, Meeyoung Cha, and Krishna P. Gummadi. 2009. On the Evolution of User Interaction in Facebook. In *Proceedings of the 2nd ACM SIGCOMM Workshop on Social Networks (WOSN)*, Barcelona, Spain.
- Ye Wang, Srinivasan Parthasarathy, and Shirish Tatikonda. 2011. Locality Sensitive Outlier Detection: A ranking driven approach. In *Proceedings of the 27th International Conference on Data Engineering (ICDE)*, Hannover, Germany. 410–421.
- Richard C. Wilson and Ping Zhu. 2008. A study of graph spectra for comparing graphs and trees. *Journal of Pattern Recognition* 41, 9 (2008), 2833–2841.
- Ömer Nebil Yaveroğlu, Noël Malod-Dognin, Darren Davis, Zoran Levnajić, Vuk Janjic, Rasa Karapandza, Aleksandar Stojmirovic, and Nataša Pržulj. 2014. Revealing the Hidden Language of Complex Networks. *Scientific reports* 4 (2014).
- Jonathan S. Yedidia, William T. Freeman, and Yair Weiss. 2003. *Understanding belief propagation and its generalizations*. Morgan Kaufmann Publishers Inc., 239–269.
- Weiren Yu, Xuemin Lin, Wenjie Zhang, Lijun Chang, and Jian Pei. 2013. More is Simpler: Effectively and Efficiently Assessing Node-Pair Similarities Based on Hyperlinks. *Proceedings of the VLDB Endowment* 7, 1 (2013), 13–24.

Received September 2013; revised month 2013; accepted month 2013