DELTACON: A Principled Massive-Graph Similarity Function with Attribution

Danai Koutra, Computer Science and Engineering, University of Michigan, Ann Arbor[1]
Neil Shah, Computer Science Department, Carnegie Mellon University.
Joshua T. Vogelstein, Department of Biomedical Engineering & Institute of Computational Medicine, Johns Hopkins University Child Mind Institute.
Brian Gallagher, Lawrence Livermore National Laboratory.
Christos Faloutsos, Computer Science Department, Carnegie Mellon University.

How much has a network changed since yesterday? How different is the wiring of 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 DELTACON, 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 DELTACON-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 DELTACON and DELTACON-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 ($\sim$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, culprit nodes and edges

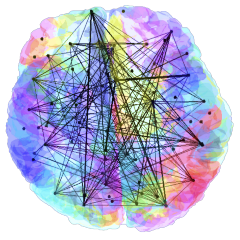
ACM Reference Format:
Danai Koutra, Neil Shah, Joshua T. Vogelstein, Brian Gallagher, and Christos Faloutsos, 2015. DELTACON: A Principled Massive-Graph Similarity Function with Attributions. *ACM Trans. Knowl. Discov. Data.* V, N, Article 0000 (2015), 38 pages.
DOI: http://dx.doi.org/10.1145/0000000.0000000

1. INTRODUCTION

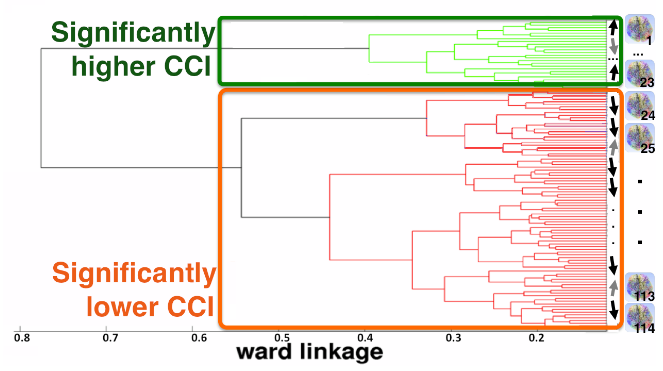
Graphs arise naturally in numerous situations; social, traffic, collaboration and computer networks, images, protein-protein interaction networks, brain connectivity graphs and web graphs are only a few examples. A problem that comes up often in all those settings is the following: how much do two graphs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

[1]Work done while at Carnegie Mellon University.

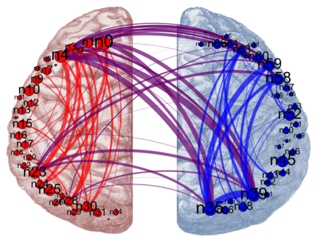
Author's address: Danai Koutra, Computer Science and Engineering, University of Michigan, Ann Arbor, email: dkoutra@umich.edu.



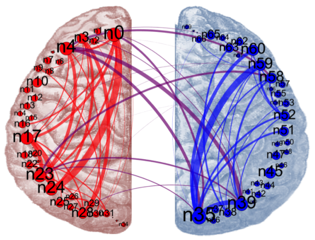
(a) Connectome: neural network of brain.



(b) Dendogram representing the hierarchical clustering of the DELTACON similarities between the 114 connectomes.



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



(d) Brain graph of a subject with low creativity index.

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) Visualization of brain graphs for subjects with high and low 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 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 the 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 [Caceres et al. 2011; Noble and Cook 2003; Wang et al. 2011; Wang et al. 2014; Koutra et al. ; Koutra et al. 2015; Shah et al. 2015].

Long in the purview of researchers, graph similarity has been 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 the 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 Figure 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 levels 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; Sections 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 Sections 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 set[2], $\mathcal{V}$, 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.

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 Figure 3 of p. 16, $(B10, mB10)$ 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. 5) that our method does exactly this in a principled way.

[2]If 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
$\mathbf{I}$	$n \times n$ identity matrix
$\mathbf{A}$	$n \times n$ adjacency matrix with elements a_{ij}
$\mathbf{D}$	$n \times n$ diagonal degree matrix, $d_{ii} = \sum_j a_{ij}$
$\mathbf{L}$	$= \mathbf{D} - \mathbf{A}$ laplacian matrix
$\mathbf{S}$	$n \times n$ matrix of final scores with elements s_{ij}
$\mathbf{S}'$	$n \times g$ 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
$\mathbf{DC}_0$, $\mathbf{DC}$	DeltaCon$_0$, DeltaCon
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 $\mathbf{A}$
λ-**D Lap**	λ-distance on the Laplacian $\mathbf{L}$
λ-**D N.L.**	λ-distance on the normalized Laplacian $\mathbf{L}$

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 matrix[3] $\mathbf{S}$. 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 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 with respect to 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:

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

[3]In reality, we don't measure all the affinities (see Section 3.2 for an efficient approximation).

where $\vec{s}_i = [s_{i1}, ...s_{in}]^T$ 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

$$\boxed{\mathbf{S} = [s_{ij}] = [\mathbf{I} + \epsilon^2\mathbf{D} - \epsilon\mathbf{A}]^{-1}}. \tag{2}$$

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 Equation 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 (Equation 1) can be written in a 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 $\mathbf{A}_* = \mathbf{D}(\mathbf{I}+\epsilon^2\mathbf{D})^{-1}\mathbf{D}^{-1}\mathbf{AD}$.

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

$[\mathbf{I} + \epsilon^2\mathbf{D} - \epsilon\mathbf{A}]\vec{s_i} = \vec{e_i}$ ($\times$ $\mathbf{D}^{-1}$ from the left)
$[\mathbf{D}^{-1} + \epsilon^2\mathbf{I} - \epsilon\mathbf{D}^{-1}\mathbf{A}]\vec{s_i} = \mathbf{D}^{-1}\vec{e_i}$ ($\mathbf{F} = \mathbf{D}^{-1} + \epsilon^2\mathbf{I}$)
$[\mathbf{F} - \epsilon\mathbf{D}^{-1}\mathbf{A}]\vec{s_i} = \mathbf{D}^{-1}\vec{e_i}$ ($\times$ $\mathbf{F}^{-1}$ from the left)
$[\mathbf{I} - \epsilon\mathbf{F}^{-1}\mathbf{D}^{-1}\mathbf{A}]\vec{s_i} = \mathbf{F}^{-1}\mathbf{D}^{-1}\vec{e_i}$ ($\mathbf{A}_* = \mathbf{F}^{-1}\mathbf{D}^{-1}\mathbf{AD}$)
$[\mathbf{I} - \epsilon\mathbf{A}_*\mathbf{D}^{-1}]\vec{s_i} = (1-\epsilon)\ (\mathbf{A}_*\mathbf{D}^{-1}\mathbf{A}^{-1}\frac{1}{1-\epsilon}\vec{e_i})$ □

2.3. Why use Belief Propagation?

The reasons we choose BP and its fast approximation with Equation 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 a 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 $sim(G_1, G_2) \in [0, 1]$ denote their similarity score. Then, we want the similarity measure to obey the following axioms:

- A1. *Identity property*: $sim(G_1, G_1) = 1$

- A2. *Symmetric property*: $sim(G_1, G_2) = sim(G_2, G_1)$
- A3. *Zero property*: $sim(G_1, G_2) \to 0$ for $n \to \infty$, 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 Equation 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[4])

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

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 affinities[5] 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.

[4]Using p^{th} root instead of square root gives consistent results (for small values of p). Without loss of generality, we use the Matusita distance, which corresponds to $p = 2$.
[5]The node affinities are in $[0, 1]$, so the square root makes them bigger.

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,∞). Notice that the distance-to-similarity transformation does *not* change the ranking of results in a nearest-neighbor query.

The straightforward algorithm, DELTACON$_0$ (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: DELTACON$_0$

```
INPUT: edge files of $G_1(\mathcal{V}, \mathcal{E}_1)$ and $G_2(\mathcal{V}, \mathcal{E}_2)$, i.e., $\mathbf{A}_1$ and $\mathbf{A}_2$
// $\mathcal{V} = \mathcal{V}_1 \cup \mathcal{V}_2$, 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 $G_1$
$d(G_1, G_2)$ =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

DELTACON$_0$ 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 DELTACON$_0$ 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.

PROOF. We can compute the $n \times g$ "skinny" matrix $\mathbf{S}'$ quickly, by solving $[\mathbf{I} + \epsilon^2\mathbf{D} - \epsilon\mathbf{A}]\mathbf{S}' = [\vec{s}_{01} \ \dots \vec{s}_{0g}]$, where $\vec{s}_{0k} = \sum_{i \in group_k} \vec{e}_i$ is the membership $n \times 1$ vector for group k (all 0s, except 1s for members of the group). Solving this system is equivalent to solving for each group g the linear system $[\mathbf{I} + \epsilon^2\mathbf{D} - \epsilon\mathbf{A}]\mathbf{S}' = \vec{s}_{0g}$. 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 $\epsilon\mathbf{A} - \epsilon^2\mathbf{D}$, 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 Equation 2. With careful implementation, DELTACON is linear on the number of edges and groups g. As we show in Section 5.3, it takes $\sim$ 160sec, on commodity hardware, for a 1.6-million-node graph. Once we have the reduced affinity matrices $\mathbf{S}'_1$ and $\mathbf{S}'_2$ 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.

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 sparse real graphs, and also the fact that changes within a group manifest less when a group consists of the nodes belonging to a single community than randomly assigned nodes.

Next we give the time complexity of DELTACON, as well as the relationship between the similarity scores of DELTACON$_0$ and DELTACON.

ALGORITHM 2: DELTACON

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

$\{\mathcal{V}_j\}_{j=1}^{g}$ = random_partition($\mathcal{V}, g$)                    //$g$ groups
// estimate affinity vector of nodes $i = 1, \ldots, n$ to group $k$
for $k = 1 \to g$ do
    $\vec{s_0}_k = \sum_{i \in \mathcal{V}_k} \vec{e}_i$
    solve $[\mathbf{I} + \epsilon^2 \mathbf{D}_1 - \epsilon \mathbf{A}_1]\vec{s}'_{1k} = \vec{s_0}_k$
    solve $[\mathbf{I} + \epsilon^2 \mathbf{D}_2 - \epsilon \mathbf{A}_2]\vec{s}'_{2k} = \vec{s_0}_k$
end for
$\mathbf{S}'_1 = [\vec{s}'_{11}\ \vec{s}'_{12}\ \ldots\ \vec{s}'_{1g}]$;  $\mathbf{S}'_2 = [\vec{s}'_{21}\ \vec{s}'_{22}\ \ldots\ \vec{s}'_{2g}]$
// compare affinity matrices $\mathbf{S}'_1$ and $\mathbf{S}'_2$
$d(G_1, G_2)$ =ROOTED ($\mathbf{S}'_1, \mathbf{S}'_2$)
return $sim(G_1, G_2) = \frac{1}{1+d(G_1,G_2)}$
```

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 Equation 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($g \cdot max\{m_1, m_2\}$). □

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 DELTACON$_0$).

PROOF. Let $\mathbf{B} = \mathbf{I} + \epsilon^2 \mathbf{D} - \epsilon \mathbf{A}$. Then DELTACON$_0$ consists of solving for every node $i \in \mathcal{V}$ the equation $\mathbf{B}\vec{s}_i = \vec{e}_i$; DELTACON solves the equation $\mathbf{B}\vec{s}'_k = \vec{s_0}_k$ for all groups $k \in (0, g]$, where $\vec{s_0}_k = \sum_{i \in group_k} \vec{e}_i$. Because of the linearity of matrix additions, it holds true that $\vec{s}'_k = \sum_{i \in group_k} \vec{s}_i$, for all groups k. □

THEOREM 2.

DELTACON's similarity score between any two graphs G_1, G_2 upper bounds the actual DELTACON$_0$'s similarity score, i.e., $sim_{DC_0}(G_1, G_2) \leq sim_{DC}(G_1, G_2)$.

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 score matrices of G_1 and G_2 by applying DELTACON$_0$, and $\mathbf{S}'_1, \mathbf{S}'_2$ be the respective $n \times g$ final score matrices by applying DELTACON. We want to show that DELTACON$_0$'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 $d^2_{DC_0} > d^2_{DC}$. 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 DELTACON$_0$, and by applying Lemma 3, we obtain the following terms:

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

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

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

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

or equivalently that $t_{DC_0} > t_{DC}$. □

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 Section 2.4.

A1. *Identity Property*: $sim(G_1, G_1) = 1$.

The affinity scores, $\mathbf{S}$, 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*: $sim(G_1, G_2) \to 0$ for $n \to \infty$, 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 $\{s_g, s_{ng}\} \to \{\frac{n}{2g}+1, \frac{n}{2g}\}$ as $n \to \infty$ (for finite $\frac{n}{g}$). Given this condition, it can be derived that the ROOTED, $d(G_1, G_2)$, between the $\mathbf{S}$ matrices of the empty and the complete graph becomes arbitrarily large. So, $sim(G_1, G_2) = \frac{1}{1+d(G_1,G_2)} \to 0$ for $n \to \infty$. □

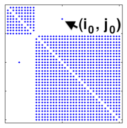
DELTACON satisfies the three axioms that every similarity measure must obey. We elaborate on the satisfiability of the properties of $P1 - P3$ next.

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.

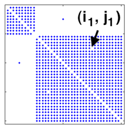
PROOF. Assume that $\mathbf{A}$ is the adjacency matrix of an undirected barbell graph with two cliques of size n_1 and n_2 respectively (e.g., $B10$ with $n_1 = n_2 = 5$ in Figure 3) and (i_0, j_0) is the "bridge" edge. Without loss of generality we can assume that $\mathbf{A}$ has a block-diagonal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

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

A = 

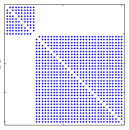
Then, $\mathbf{B}$ is an adjacency matrix with elements

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

B = 

and $\mathbf{C}$ 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 = 

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

From Equation 1, by expressing the matrix inversion using a power series and ignoring the terms of greater than second power, for matrix $\mathbf{A}$ we obtain the solution:

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

where the index (e.g., $\mathbf{A}$) denotes the graph each matrix corresponds to. Now, using the last equation, we can write out the elements of the $\mathbf{S_A}, \mathbf{S_B}, \mathbf{S_C}$ 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 the 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 $c1 \geq 2\sqrt{(\epsilon)}$ and $c2 \geq \sqrt{\epsilon}$ for $n_2 \geq 3$ and using these facts in the last equation, it follows that:

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

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 $a_{i_0j_0} = 1$, 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 $\mathbf{H}$[6] 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 $sim(\mathbf{A}, \mathbf{B}) \geq sim(\mathbf{C}, \mathbf{H}) \Leftrightarrow d(\mathbf{A}, \mathbf{B}) \leq d(\mathbf{C}, \mathbf{H})$. 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 Equation 1 using a power series and ignoring the terms of greater than second power:

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

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 distance since that function is mathematically easier to manipulate than the similarity function. In Figure 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 $(\mathbf{S}_B)_{ij} \geq (\mathbf{S}_A)_{ij}$.

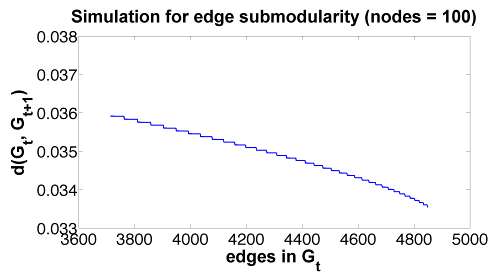
PROOF OF LEMMA 4. From Equation 1, by expressing the matrix inversion using a power series and ignoring the terms of greater than second power, we obtain the solution:

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

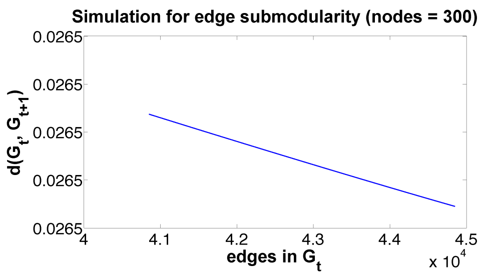
or equivalently

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

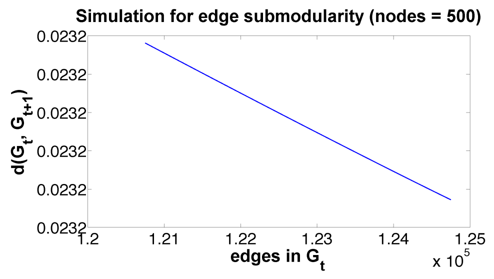
[6]We use $\mathbf{H}$ instead of $\mathbf{D}$ to distinguish the adjacency matrix of the graph from the diagonal matrix of degrees, which is normally defined as $\mathbf{D}$.



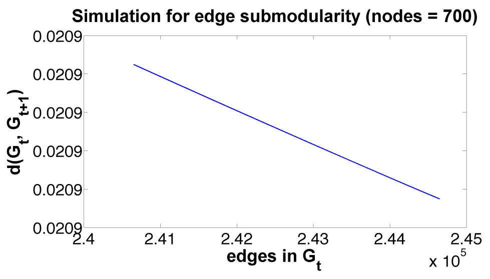
(a) Graph of size 100



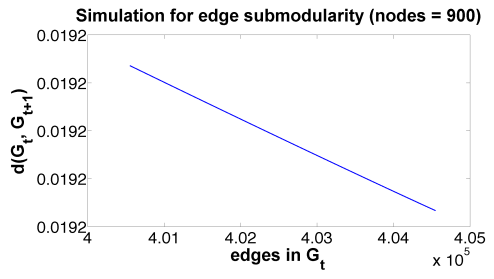
(b) Graph of size 300



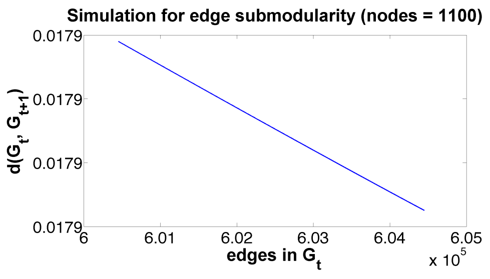
(c) Graph of size 500



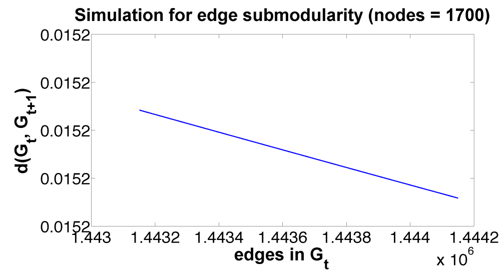
(d) Graph of size 700



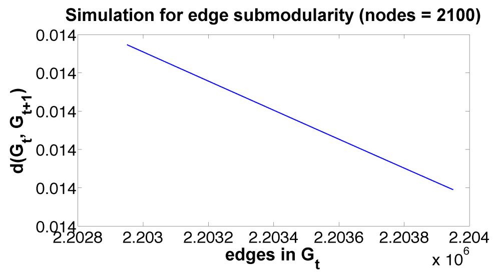
(e) Graph of size 900



(f) Graph of size 1100



(g) Graph of size 1700



(h) Graph of size 2100

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.

and

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

Note that

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

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 Equation 4, we have 3 cases:

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

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

— 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 $(\mathbf{S}_B)_{ij} \geq (\mathbf{S}_A)_{ij}$. □

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 $sim(\mathbf{A}, \mathbf{B}) \geq sim(\mathbf{A}, \mathbf{C}) \Leftrightarrow d(\mathbf{A}, \mathbf{B}) \leq d(\mathbf{A}, \mathbf{C})$. 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 $(\mathbf{S}_B)_{ij} \geq (\mathbf{S}_\mathbf{A})_{ij}$, $(\mathbf{S}_\mathbf{C})_{ij} \geq (\mathbf{S}_\mathbf{A})_{ij}$, and $(\mathbf{S}_\mathbf{C})_{ij} \geq (\mathbf{S}_\mathbf{B})_{ij}$ for all i, j by the construction of the matrices $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{C}$, and Lemma 4 . □

In Section 5, we show experimentally that DELTACON not only 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 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 $\mathbf{A}$ and the affinity of node v to the node groups in graph $\mathbf{A_2}$. 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 $\mathbf{S}_1'$ and $\mathbf{S}_2'$ 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 $\mathbf{S}_1'$ and $\mathbf{S}_2'$ as w_v for $v = 1, \dots, n$ per Equation 5.

$$w_v = \text{ROOTED}(\mathbf{S}_{1,v}', \mathbf{S}_{2,v}') = \sqrt{\sum_{j=1}^{g} (\sqrt{s_{1,vj}'} - \sqrt{s_{2,vj}'})^2}. \tag{5}$$

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 $\mathbf{S}_1', \mathbf{S}_2'$
       edge files of $G_1(\mathcal{V}, \mathcal{E}_1)$ and $G_2(\mathcal{V}, \mathcal{E}_2)$, i.e., $\mathbf{A}_1$ and $\mathbf{A}_2$

for $v = 1 \to n$ do
    // If an edge adjacent to the node has changed, the node is responsible:
    if $\sum |\mathbf{A}_1(v,:) - \mathbf{A}_2(v,:)| > 0$ then
        $w_v$ = ROOTED$(\mathbf{S}_{1,v}', \mathbf{S}_{2,v}')$
    end if
end for
$[\vec{w}_{sorted}, \vec{w}_{sortedIndex}]$ = sortRows($\vec{w}$, 1, 'descend')    // sort rows of vector $w$ on column index 1
                                                                    //(node impact score) by descending value
return $[\vec{w}_{sorted}, \vec{w}_{sortedIndex}]$
```

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 an impact score. This score is equal to the sum of impact scores for the nodes that the edge 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 are indicative of community expansion or reduction via addition or removal of certain key bridge edges.

ALGORITHM 4: DELTACON-ATTR Edge Attribution

```
INPUT: adjacency matrices $\mathbf{A}_1$, $\mathbf{A}_2$, culprit set of interest $\vec{w}_{sortedIndex,1\ldots\text{index}}$ and node impact scores $\vec{w}$

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

4.2. Scalability Analysis

Given precomputed $\mathbf{S}_1'$ and $\mathbf{S}_2'$ (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 only need to 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 (Figure 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 Equation 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 Figure 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 informal property (IP), we use real networks with up to 11 million edges (Table VI, page 19).

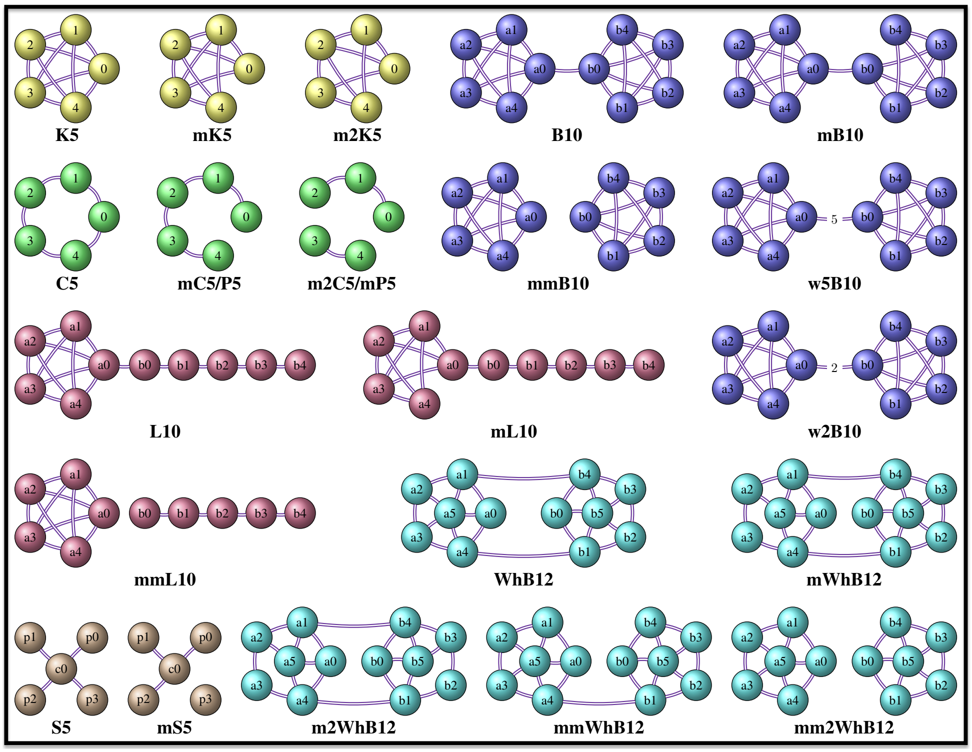


Fig. 3: Small, synthetic graphs used in DELTACON experimental analysis – *K: clique, C: cycle, P: path, S: star, B: barbell, L: lollipop, and WhB: wheel-barbell.*

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

Symbol	Meaning	Symbol	Meaning
$\mathbf{K_n}$	clique of size n	$\mathbf{B_n}$	barbell of size n
$\mathbf{P_n}$	path of size n	$\mathbf{WhB_n}$	wheel barbell of size n
$\mathbf{C_n}$	cycle of size n	$\mathbf{m_X}$	missing X edges
$\mathbf{S_n}$	star of size n	$\mathbf{mm_X}$	missing X "bridge" edges
$\mathbf{L_n}$	lollipop of size n	$\mathbf{w}$	weight of "bridge" edge

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

(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:

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

(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].

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

For $\mathcal{V}_1 = \mathcal{V}_2$ and unweighted graphs, sim_{GED} is equivalent to the Hamming distance (HD) defined as HD$(\mathbf{A_1}, \mathbf{A_2}) = sum(\mathbf{A_1}\, XOR\, \mathbf{A_2})$.

(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 Figure 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 the role of the edge is.

OBSERVATION 2. Only DELTACON succeeds in distinguishing the importance of the edges ($P1$) with respect to 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 $m_xA(\mathcal{V}, \mathcal{E}_1)$ and $m_xB(\mathcal{V}, \mathcal{E}_2)$ are the respective derived graphs after removing x edges. We expect that $sim(A, m_xA) > sim(B, m_xB)$, since the fewer the edges in a constant-sized graph, the more "important" they are."

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 the violation of the "edge-submodularity" property.

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	$\Delta s = sim(A,B) - sim(A,C)$				$\Delta d = d(A,C) - d(A,B)$			
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	$\Delta s = sim(A,B) - sim(C,D)$				$\Delta d = d(C,D) - d(A,B)$			
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}	$\mathbf{10^{-4}}$	**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	$\Delta s = sim(A,B) - sim(C,D)$				$\Delta d = d(C,D) - d(A,B)$			
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: DELTACON$_0$ and DELTACON (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 $\Delta s = sim(A,B) - sim(C,D)$ 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.

OBSERVATION 3. Only DELTACON 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 Figure 4, for different levels of corruption (10%, 20%, ..., 80%), we give 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 ≈ targeted."* In Figure 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).

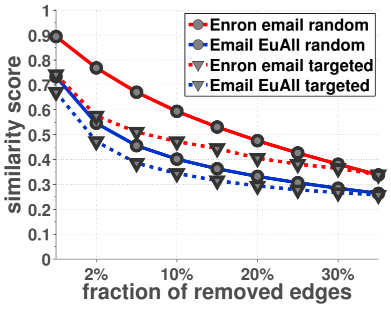
This is expected, because the random and targeted edge removal tend to be equivalent when a significant fraction of edges is deleted.

In Figures 4(a)-(f), 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.

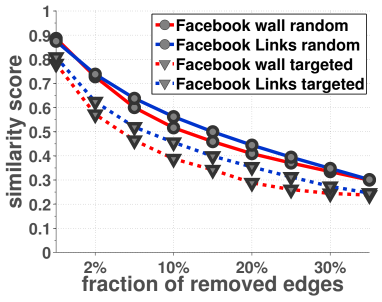
OBSERVATION 6. *"More changes hurt more."* The higher the corruption of the original graph, the smaller the DELTACON similarity between the derived and the original graph is. In Figures 4(a)-(f), we observe that as the percentage of removed edges increases, the similarity to the original graph decreases consistently for a variety of real graphs.

Table VI: Large Real and Synthetic Datasets

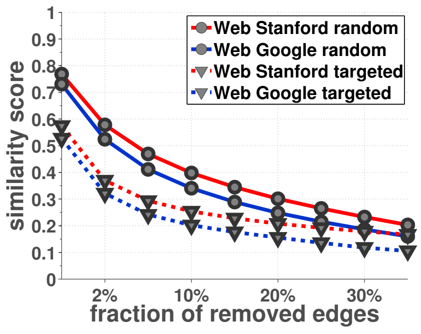
Name	Nodes	Edges	Description
Brain Graphs Small [OCP 2014]	70	800-1,208	connectome
Enron Email [Klimt and Yang 2004]	36,692	367,662	who-emails-whom
Facebook wall [Viswanath et al. 2009]	45,813	183,412	who-posts-to-whom
Facebook links [Viswanath et al. 2009]	63,731	817,090	friend-to-friend
Epinions [Guha 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 [SNAP]	325,729	1,497,134	site-to-site
Web Stanford [SNAP]	281,903	2,312,497	site-to-site
Web Google [SNAP]	875,714	5,105,039	site-to-site
Web Berk/Stan [SNAP]	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



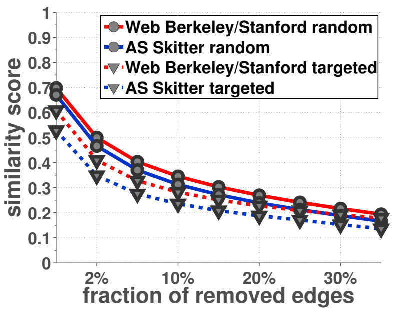
(a) Email Networks



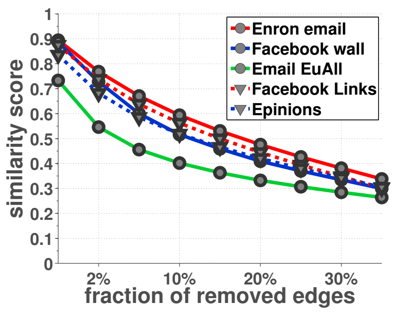
(b) Facebook Networks



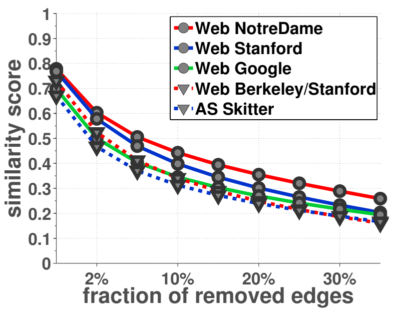
(c) Web Stanford and Google



(d) Web Berkeley/Stanford and AS Skitter



(e) Social networks.



(f) Web and p2p graphs.

Fig. 4: DeltaCon is "focus-aware" (IP): Targeted changes hurt more than random ones. (a)-(f): DeltaCon similarity scores for random (solid lines) and targeted (dashed lines) changes vs. the fraction of removed edges in the "corrupted" versions of the original graphs (x axis). We note that the dashed lines are always below the solid lines of the same color. (e)-(f):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.

General Remarks. All in all, the baseline methods have several non-desirable properties. The spectral methods, as well as SS fail to comply with the "edge importance" (P1) and "edge submodularity" (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 to 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.

5.2. Intuitiveness of DELTACON-ATTR

In addition to evaluating the intuitiveness of DELTACON$_0$ and DELTACON, we also test DELTACON-ATTR on a number of synthetically created and modified graphs, and compare it to the state-of-the-art methods. We perform two types of experiments: The first experiment examines whether the *ranking* of the culprit nodes by our method agrees with intuition. In the second experiment, we evaluate DELTACON-ATTR's *classification* accuracy in finding culprits, and compare it to the best-performing competitive approach, CAD[7], which was introduced by Sricharan and Das [Sricharan and Das 2014] concurrently, and independently from us. CAD uses the idea of commute time between nodes to define the anomalousness of nodes/edges. In a random walk, the commute time is defined as the expected number of steps starting at i, before node j is visited and then node i is reached again. We give a qualitative comparison between DELTACON-ATTR and CAD in Section 7 (Node/Edge Attribution).

Ranking Accuracy. We extensively tested DELTACON-ATTR on a number of synthetically created and modified graphs, and compared it with CAD. We note that CAD was designed to simply identify culprits in time-evolving graphs without ranking them. In order to compare it with our method, we adapted CAD so that it returns ranked lists of node and edge culprits: (i) We rank the culprit edges in decreasing order of edge score ΔE; (ii) To each node v, we attach a score equal to the sum of the scores of its adjacent edges, i.e., $\sum_{u \in N(v)} \Delta E((v,u))$, where $N(v)$ are the neighbors of v. Subsequently, we rank the nodes in decreasing order of attached score.

We give several of the conducted experiments in Table VII, and the corresponding graphs in Figures 5 and 6. Each row of the table corresponds to a comparison between graph A and graph B. The node and edge culprits that explain the main differences between the compared graphs are annotated in Figures 5 and 6. The darker a node is, the higher it is in the ranked list of node culprits. Similarly, edges that are adjacent to darker nodes are higher in the ranked list of edge culprits than edges that are adjacent to lighter nodes. If the returned ranked list agrees with the expected list (according to the formal and informal properties), we characterize the

[7]CAD was originally introduced for finding culprit nodes and edges without ranking them. We extended the proposed method to rank the culprits.

Table VII: DELTACON-ATTR obeys all the required properties. Each row corresponds to a comparison between graph A and graph B, and evaluates the node and edge attribution of DELTACON-ATTR and CAD. The right order of edges and nodes is marked in Figures 5 and 6. We give the ranking of a method if it is different from the expected one.

Graph A	Graph B	DELTACON-ATTR edges	DELTACON-ATTR nodes	CAD edges: δ for l = 5	CAD nodes	Properties
K5	mK5	✔	✔	✔	✔	
K5	m2K5	✔	✔	✔	✔	IP
B10	mB10 ∪ mmB10	✔	✔	✔	✔	P1, P2, IP
L10	mL10 ∪ mmL10	✔	✔	✔	5,6,4	P1, IP
S5	mS5	✔	✔	✔	1=5	P1 , P2
K100	mK100	✔	✔	✔	✔	
K100	w5K100	✔	✔	✔	✔	P3
mK100	w5K100	✔	✔	✔	✔	P3
K100	m3K100	✔	✔	✔	✔	P3, IP
K100	m10K100	✔	✔	(80,82)=(80,88)=(80,92)*	80,30,88=92*	P3, IP
P100	mP100	✔	✔	✔	✔	P1
w2P100	w5P100	✔	✔	✔	✔	P1, P3
B200	mmB200	✔	✔	✔	✔	P1
w20B200	m3B200	✔	✔	✔	✔	P1, P3, IP
S100	mS100	✔	✔	✔	1=4	P1, P2
S100	m3S100	✔	✔	✔	1,81=67=4	P1, P2, IP
wS100	m3S100	✔	✔	(1,4),(1,67),(1,81)	1,4=67,81	P1, P3, IP
Custom18	m2Custom18	✔	✔	(18,17),(10,11)	18,17,10,11	P1, P2
Custom18	m4Custom18b	✔	✔	✔	5=6,17=18	P1, P3

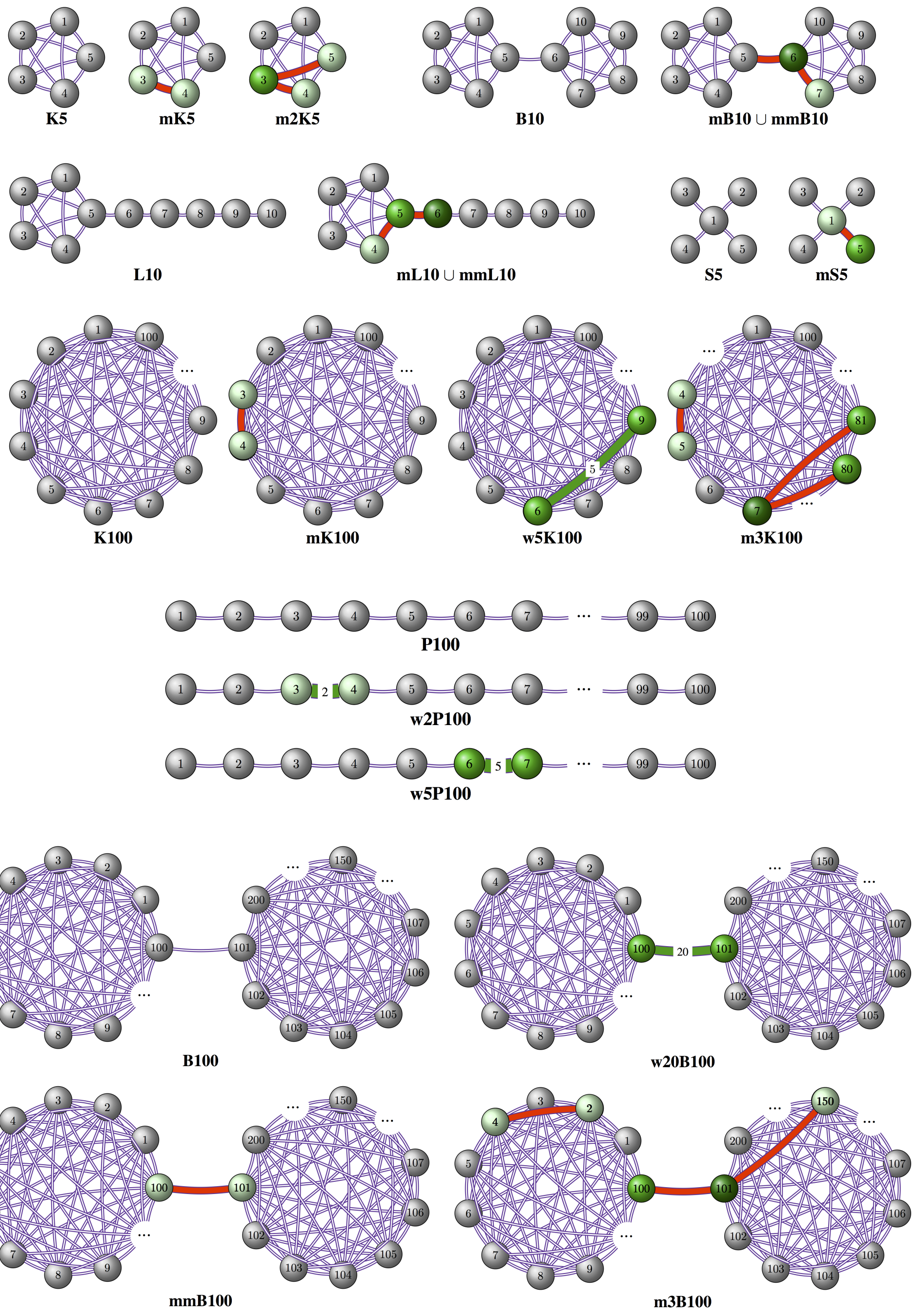


Fig. 5: DELTACON-ATTR respects properties P1-P3, and IP. Nodes marked green are identified as the culprits for the change between the graphs. Darker shade corresponds to higher rank in the list of culprits. Removed and weighted edges are marked red and green, respectively.

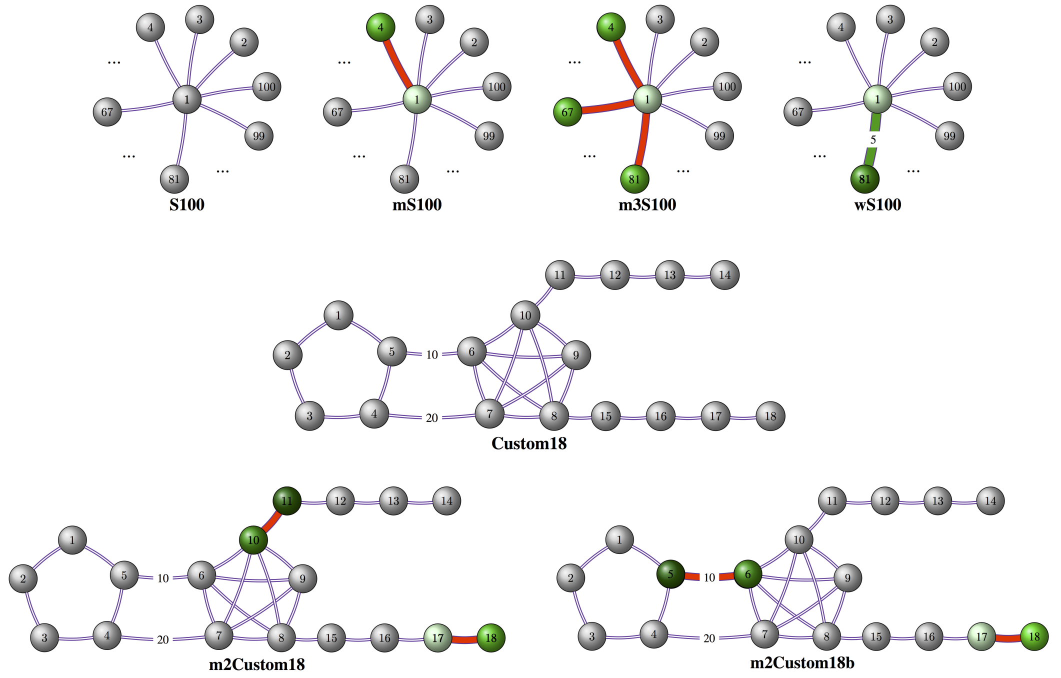


Fig. 6: [continued] DELTACON-ATTR respects properties P1-P3, and IP. Nodes marked green are identified as the culprits for the change between the graphs. Darker shade corresponds to higher rank in the list of culprits. Removed edges are marked red.

attribution of the method correct (checkmark). If there is disagreement, we provide the ordered list that the method returned. If two nodes or edges are tied, we use "=". For CAD we picked the parameter δ such that the algorithm returns 5 culprit edges and their adjacent nodes. Thus, we mark the returned list with "*" if CAD outputs 5 culprits while more exist. For each comparison, we also give the properties (last column) that define the order of the culprit edges and nodes.

OBSERVATION 7. DELTACON-ATTR reflects the desired properties P1, P2, P3, and IP, while CAD fails to return the expected ranked lists of node and edge culprits in several cases.

Next we explain some of the comparisons that we present in Table VII:

- **K5-mK5**: The pair consists of a 5-node complete graph and the same graph with one missing edge, $(3, 4)$. DELTACON-ATTR considers nodes 3 and 4 top culprits, with equal rank, due to equivalent loss in connectivity. Edge $(3, 4)$ is ranked top, and is essentially the only changed edge. CAD finds the same results.
- **K5-m2K5**: The pair consists of a 5-node complete graph and the same graph with two missing edges, $(3, 4)$ and $(3, 5)$. Both DELTACON-ATTR and CAD consider node 3 the top culprit, because two of its adjacent edges were removed. Node 3 is followed by 4 and 5, which are tied since they are both missing one adjacent edge (Property IP). The removed edges, $(3, 4)$ and $(3, 5)$, are considered equally responsible for the difference between the two input graphs. We observe similar behavior in larger complete graphs with 100 nodes (K100, and modified graphs mK100, w5K100 etc.). In the case of K100 and m10K100[8],

[8]m10K100 is a complete graph of 100 nodes where we have removed 10 edges: (i) 6 of the edges were adjacent to node 80—$(80, 82), (80, 84), (80, 86), (80, 88), (80, 90), (80, 92)$; (ii) 3 of the edges were adjacent to node 30—$(30, 50), (30, 60), (30, 70)$; and (iii) edge $(1, 4)$.

CAD does not return all 13 node culprits and 10 culprit edges because its parameter, δ, was set so that it would return at most 5 culprit edges[9].

- **B10-mB10 $\cup$ mmB10**: We compare a barbell graph of 10 nodes to the same graph that is missing both an edge from a clique, $(6, 7)$, and the bridge edge, $(5, 6)$. As expected, DELTACON-ATTR finds 6, 5 and 7 as top culprits, where 6 is ranked higher than 5, since 6 lost connectivity to both nodes 5 and 7, whereas 5 disconnected only from 6. Node 5 is ranked higher than 7 because the removal of the bridge edge is more important than the removal of $(6, 7)$ within the second clique (Property P1). CAD returns the same results. We observe similar results in the case of the larger barbell graphs (B200, mmB200, w20B200, m3B200).
- **L10-mL10 $\cup$ mmL10**: This 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 6, 5 and 4 are considered the top culprits for the difference in the graphs. Moreover, 6 is ranked more responsible for the change than 5, since 6 lost connectivity to a more strongly connected component than 5 (Property P2). However, CAD ranks node 5 higher than node 6 despite the difference in the connectivity of the two components (violation of P2).
- **S5, mS5**: We compare a 5-node star graph, and the same graph missing the edge $(1, 5)$. DELTACON-ATTR considers 5 and 1 top culprits, with 5 ranking higher than 1, as the edge removal caused a loss of connectivity from node 5 to all the peripheral nodes of the star, $2, 3, 4$, and the central node, 1. CAD considers nodes 1 and 5 equally responsible, ignoring the difference in the connectivity of the components (violation of P2). Similar results are observed in the comparisons between the larger star graphs–S100, mS100, m3S100, wS100.
- **Custom18-m2Custom18**: The ranking of node culprits that DELTACON-ATTR finds is 11, 10, 18, and 17. The nodes 11 and 10 are considered more important than the nodes 18 and 17, as the edge removal $(10, 11)$ creates a large connected component and a small chain of 4 nodes, while the edge removal $(17, 18)$ leads to a single isolated node (18). Node 10 is higher in the culprit list than node 11 because it loses connectivity to a denser component. The reasoning is similar for the ranking of nodes 18 and 17. CAD does not consider the differences in the density of the components, and leads to a different ranking of the nodes.
- **Custom18-m2Custom18**: The ranking of node culprits that DELTACON-ATTR returns is 5, 6, 18, and 17. This is in agreement with properties P1 and P3, since the edge $(5, 6)$ is more important than the edge $(17, 18)$. Node 5 is more responsible than node 6 for the difference between the two graphs, as node 5 ends up having reduced connectivity to a denser component. This property is ignored by CAD, which thus results in different node ranking.

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.

Classification Accuracy. To further evaluate the accuracy of DELTACON-ATTR in classifying nodes as culprits, we perform a simulation-based experiment and compare our method to CAD. Specifically, we set up a simulation similar to the one that was introduced in [Sricharan and Das 2014].

We sample 2000 points from a 2-dimensional Gaussian mixture distribution with four components, and construct the matrix $\mathbf{P} \in \mathcal{R}^{2000\times 2000}$, with entries $p(i, j) = \exp ||i - j||$, for each pair of points (i, j). Intuitively, the adjacency matrix $\mathbf{P}$ corresponds to a graph with four clusters that have strong connections within them, but weaker connections across them. By following the same process and adding some noise in each component of the mixture model, we also build a matrix $\mathbf{Q}$, and add more noise to it, which is defined as:

$$\mathbf{R}_{ij} = \begin{cases} 0 & \text{with probability } 0.95 \\ u_{ij} \sim \mathcal{U}(0,1) & \text{otherwise,} \end{cases}$$

where $\mathcal{U}(0, 1)$ is the uniform distribution in $(0, 1)$. Then, we compare the two graphs, G_A and G_B, to each other, which have adjacency matrices $\mathbf{A} = \mathbf{P}$ and $\mathbf{B} = \mathbf{Q} + (\mathbf{R} + \mathbf{R}')/2$, respectively. We consider culprits (or anomalous) the inter-cluster edges for which $\mathbf{R}_{ij} \neq 0$, and the adjacent nodes. According to property P1, these edges are considered important (major culprits) for the difference between the graphs, as they establish more connections between loosely coupled clusters.

Conceptually, DELTACON-ATTR and CAD are similar because they are based on related methods [Koutra et al. 2011] (Belief Propagation and Random Walk with Restarts, respectively). As shown in Figure 7, the

[9]The input graphs are symmetric. If edge (a, b) is considered culprit, CAD returns both (a, b) and (b, a), which have the same anomalousness score.

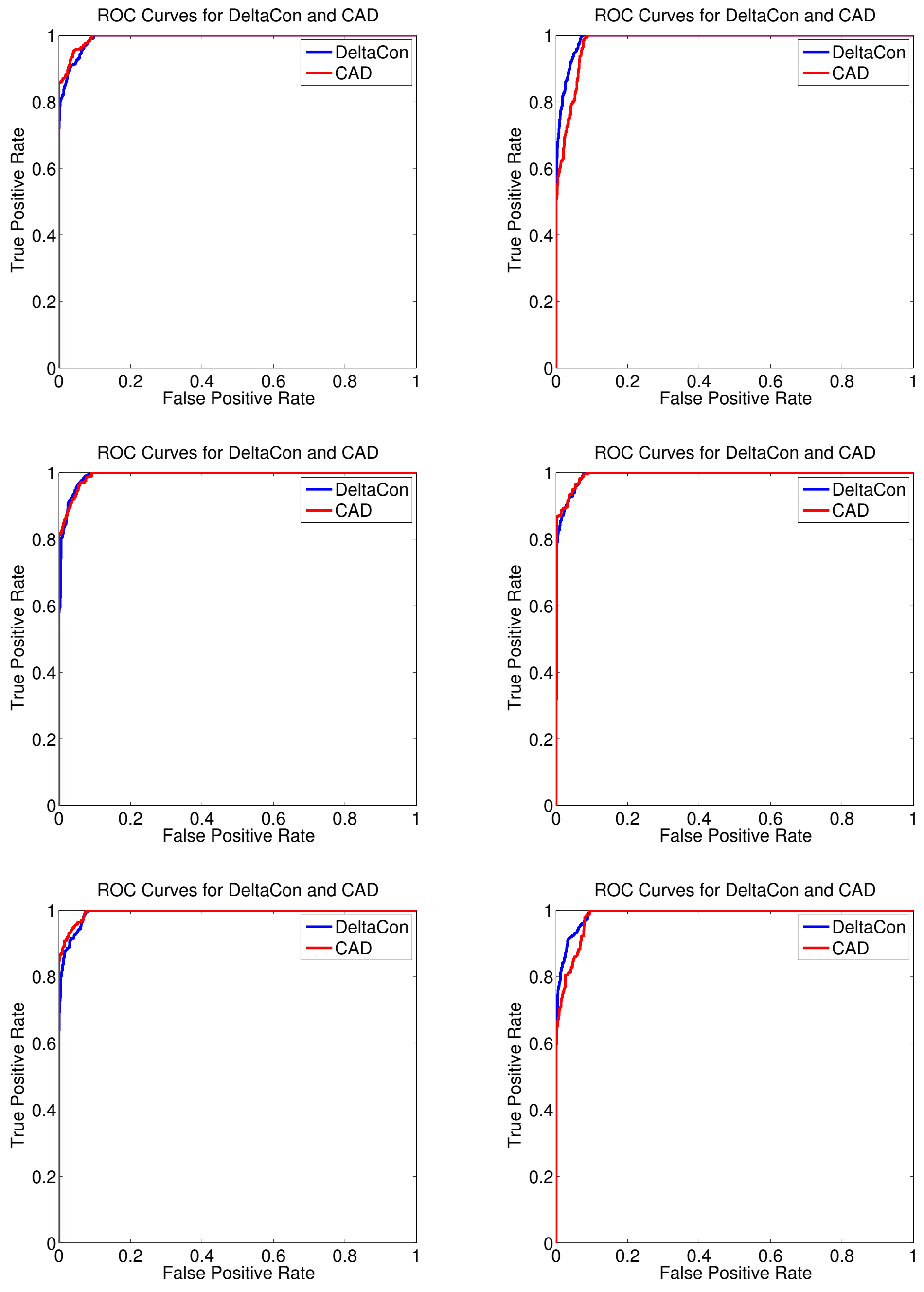


Fig. 7: DELTACON-ATTR ties state-of-the-art method with respect to accuracy. Each plot shows the ROC curves for DELTACON-ATTR and CAD for different realizations of two synthetic graphs. The graphs are generated from points sampled from a 2-dimensional Gaussian mixture distribution with four components.

simulation described above corroborates this argument, and the two methods have comparable performance – i.e., the areas under the ROC curves are similar for various realizations of the data described above. Over 15 trials, the AUC of DELTACON-ATTR and CAD is 0.9922 and 0.9934, respectively.

OBSERVATION 8. Both methods are very accurate in detecting nodes that are responsible for the differences between two highly-clustered graphs (Property P1).

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 9. As shown in Figure 8, DELTACON scales linearly with the number of edges in the largest input graph.

We note 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 already in memory from the DELTACON similarity computation. Specifically, node and edge attribution are loglinear on the nodes and edges, respectively, given that sorting is unavoidable for the task of ranking.

To compare the runtime of DELTACON-ATTR with the runtime of CAD, we perform the runtime experiment that the authors ran in [Sricharan and Das 2014]. Specifically, we generate sparse uniformly distributed random and symmetric matrices of $n = O(10^7)$ nodes, and sparsity level $\frac{1}{n}$. Over 10 trials, the combined runtime of DELTACON and DELTACON-ATTR is 4.01 minutes on average on a less powerful machine[10]. than the one used in [Sricharan and Das 2014], while the reported time for CAD is 5 minutes on average. Thus, our method is faster than CAD, while having comparable or better accuracy on small and large synthetic datasets.

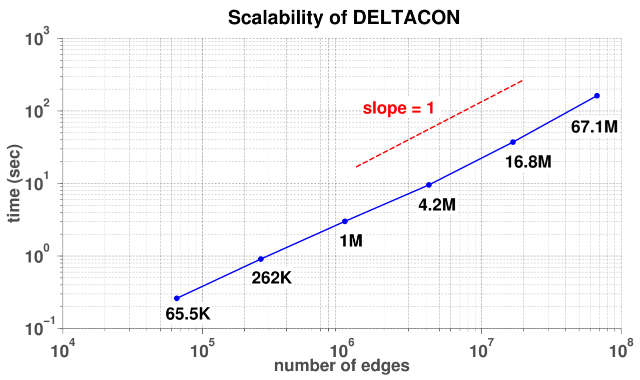
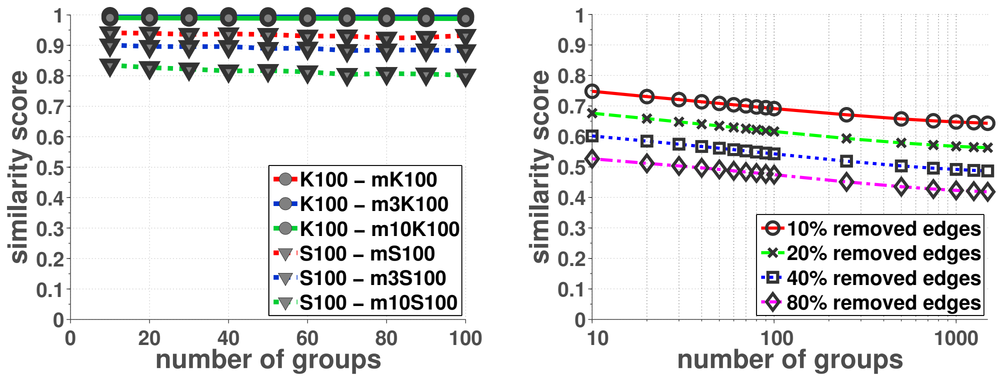


Fig. 8: DELTACON is linear on the number of edges (time in seconds vs. number of edges). The exact number of edges is annotated.

[10]The source code of [Sricharan and Das 2014] is not yet available, while we have implemented their naïve algorithm which is cubic. To make a more fair comparison between the two methods, we reproduced the experiment that the authors described, ran only our method, and compared the runtime of DELTACON to the reported runtime of CAD. For this experiment we used a 64-bit 2.8GHz *single* quad core AMD Opteron (tm) Processor 854 with 32GB RAM, while the authors of CAD used a 64-bit 2.3 GHz *dual* quad core Dell Precision T7500 desktop with 32GB RAM.

5.4. Robustness of DELTACON

DELTACON$_0$ 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 DELTACON$_0$ compare.



(a) Robustness of DeltaCon on synthetic graphs. (b) Robustness of DeltaCon on the Political Blogs dataset.

Fig. 9: 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., $sim(K100, m1K100) > sim(K100, m3K100) > \ldots > sim(S100, m1S100) > \ldots > sim(S100, m10S100)$).

For this experiment, we use complete and star graphs, as well as the Political Blogs dataset. For each of the synthetic graphs (a 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 to DELTACON$_0$. The results for the synthetic and real graphs are shown in Figures 9(a) and (b), respectively.

OBSERVATION 10. In our experiments, DELTACON$_0$ and DELTACON agree on the ordering of pairwise graph similarities.

In Figure 9(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 the different graph pairs remain the same. In other words, the ordering of similarities is the same at every group level g.

OBSERVATION 11. 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 DELTACON$_0$. For instance, in Figure 9(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 DELTACON$_0$. However, even with few groups, the approximation of DELTACON$_0$ 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.

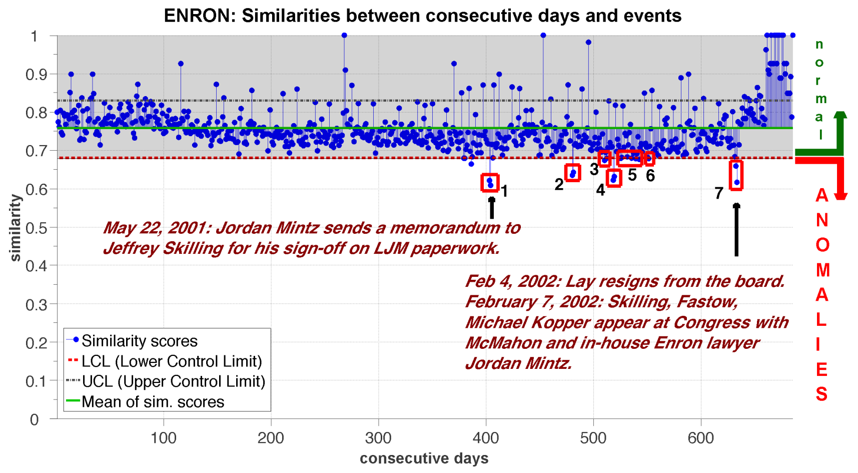


Fig. 10: DELTACON detects anomalies on Enron data coinciding with major events. The marked days correspond to anomalies. The blue points are similarity scores between consecutive instances of the daily email activity between the employees, and the marked days are 3σ units away from the median similarity score.

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 10 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 in-control similarity scores. These limits correspond to median $\pm 3\sigma$[11]. 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 Figure 10):

(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

[11]The 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.

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 .

May 2001:

- Top Influential Culprit: John Lavorato, the former head of Enron's trading operations and CEO of Enron America, connected to $\sim$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, Fift 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 L. Taylor and L. Kitchen.

6.2. Brain Connectivity Graph Clustering

We also use DELTACON for the clustering and classification of graphs. For this purpose we study *conectomes*, 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, etc. 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 one 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 (Figure 1(b)). As shown in the figure, there are two clearly separable groups of brain graphs. Applying a t-test on the available attributes for the two groups created by the clusters, we have 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 11 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 their hemispheres than those subjects that are less creative. Moreover, the two groups

(a) Subject with high creativity index. (b) Subject with low creativity index.

Fig. 11: Illustration of brain graphs for subjects with high and low creativity index, respectively. The low-CCI brain has fewer and lighter cross-hemisphere connections than the high-CCI brain.

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 the t-test when more than 2 groups are analyzed), we tested whether the various clusters that we obtain from the connectivity-based hierarchical clustering 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 brain connectivity.

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 $\sim$ 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 were 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 $C_i \in \{1, \ldots, 2n\}$, we choose the connectome $C_j \in \{1, \ldots, 2n\} \setminus i$ so that $sim(C_i, C_j)$ 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 $i \in \{1, \ldots, 2n\}$, we choose

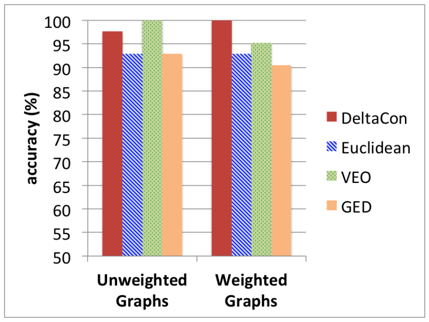
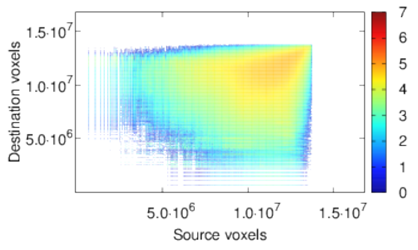
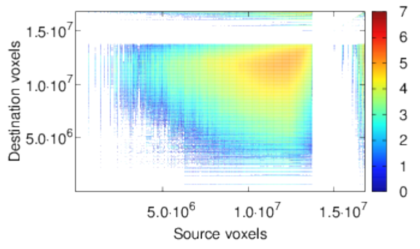


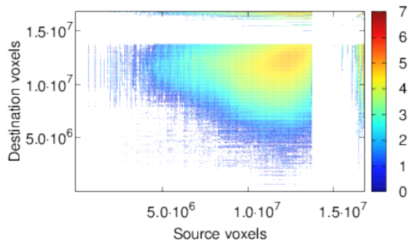
Fig. 12: DELTACON similarity outperforms almost all the baseline approaches. In the figure, the accuracy reflects the percentage of correctly recovered pairs of brain graphs. DELTACON recovers correctly *all* pairs of connectomes that correspond to the same subject for *weighted* graphs outperforming 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.



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



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



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

Fig. 13: DELTACON outperforms the Euclidean distance in recovering the correct test-retest pairs of brain graphs. Spy plots of three brain graphs (Nodes aligned in all plots, and in increasing order of degrees in plot (a).). 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 the Euclidean distance found.

the connectome $j \in \{1, \ldots, 2n\} \setminus i$ so that $ED(i, j)$ is minimized[12], 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 Figure 12.

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 (Figure 12, '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 Figure 12 (labeled 'Weighted Graphs'), while Figure 13 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 Figure 12, 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

[12]We 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 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).

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, and mention what sets our method apart.

Graph Similarity. Graph similarity refers to the problem of quantifying how similar two graphs are. The graphs may be aligned or not, which leads to two instances of the problem:

(1) *Known Node Correspondence.* The first instance of the problem 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 Signature Similarity (SS), which is based on the SimHash algorithm, while Vertex/Edge Overlap similarity (VEO) also 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 the 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. 2003]. 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 into three main approaches: (a) feature extraction and similarity computation based on the feature space, (b) graph matching and the application of techniques from the first category, and (c) graph kernels.

There are numerous works that 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 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, number 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: leadership, bonding and diversity. The complexity of the last 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 Gärtner 2003; Mahé and Vert 2009], and 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 the 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, FABP [Koutra et al. 2011], which is also intuitive. LINBP [Gatterbauer et al. 2015] generalizes FABP to handle multiple node classes, as well as a mix of network effects (e.g., heterophily and homophily). 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; Koutra et al. 2013; 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 nodes behave differently –, and also spot nodes that are most 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 –which is based on [Ide and Kashima 2004]– extracts node features, computes an 'eigen'-behavior per node, and spots changes each node's behavior over time. Therefore, the method relies on the selection of features (*e.g.,* in-degree, out-degree, edge weights, number of triangles). Moreover, because of the focus on local egonet features, it may not distinguish between small and large changes in time-evolving networks, and it also tends to return a large number of false positives [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 foundation of our method) and Random Walks with Restarts (the core idea behind 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 then defines the anomalousness of edges as a function of the outlierness of the adjacent nodes; CAD first identifies the most anomalous edges, and then defines all their adjacent nodes as anomalous 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).

In Table VIII we summarize the differences between our proposed methods and alternative approaches.

Table VIII: Qualitative comparison of DELTACON with state-of-the-art approaches.

	Graph distance	Properties P1-P3, IP	Attribution	Ranked culprits	Scalable	Parameter-free
Vertex/Edge Overlap [Papadimitriou et al. 2008]	✓	✗	✗	✗	✓	✓
Graph Edit Distance [Bunke et al. 2006]	✓	✗	✗	✗	?	✓
Signature Similarity [Papadimitriou et al. 2008]	✓	✗	✗	✗	✓	✓
λ-distance [Bunke et al. 2006; Peabody 2003; Wilson and Zhu 2008]	✓	✗	✗	✗	✗	?
CAD [Sricharan and Das 2014]	possible	✗	✓	possible	✓	✗
DELTACON DELTACON-ATTR	✓	✓	✓	✓	✓	✓

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 Section 2), (b) *intuitive* (properties $P1$-$P4$, in Section 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 the anonymous reviewers, 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 Contracts DE-AC52-07NA27344, N66001-15-C-4041, DARPA (SIMPLEX) and DARPA (GRAPHS). 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 International Conference on Management of Data (SIGMOD), Scottsdale, AZ.* 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.

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.

Keith Andrews, Martin Wohlfahrt, and Gerhard Wurzinger. 2009. Visual Graph Comparison. In *13th International Conference on Information Visualization - Showcase (IV)*. 62 –67.

Mohsen Bayati, David F. Gleich, Amin Saberi, and Ying Wang. 2013. Message-Passing Algorithms for Sparse Network Alignment. *ACM Transactions on Knowledge Discovery from Data* 7, 1, Article 3 (Helen Martin 2013), 31 pages.

Michele Berlingerio, Danai Koutra, Tina Eliassi-Rad, and Christos Faloutsos. 2013. Network Similarity via Multiple Social Theories. *IEEE/ACM Conference on Advances in Social Networks Analysis and Mining (ASONAM'13)* (2013).

Alex Beutel, Wanhong Xu, Venkatesan Guruswami, Christopher Palow, and Christos Faloutsos. 2013. CopyCatch: Stopping Group Attacks by Spotting Lockstep Behavior in Social Networks. In *Proceedings of the 22nd International Conference on World Wide Web (WWW), Rio de Janeiro, Brazil*. ACM, 119–130.

Karsten M. Borgwardt and Hans-Peter Kriegel. 2005. Shortest-Path Kernels on Graphs. In *Proceedings of the 5th IEEE International Conference on Data Mining (ICDM'05)*. IEEE Computer Society, Washington, DC, USA, 74–81.

S. Bradde, Alfredo Braunstein, H. Mahmoudi, F. Tria, Martin Weigt, and Riccardo 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 and ISDN Systems* 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 Data Mining Workshops (ICDMW) at the 11th IEEE International Conference on Data Mining (ICDM), Vancouver, Canada*. 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. *Journal of Machine Learning Research* 10 (June 2009), 747–776.

Donatello Conte, Pasquale Foggia, Carlo Sansone, and Mario Vento. 2004. Thirty Years Of Graph Matching In Pattern Recognition. *International Journal of Pattern Recognition and Artificial Intelligence* 18, 3 (2004), 265–298.

Fabrizio Costa and Kurt De Grave. 2010. Fast Neighborhood Subgraph Pairwise Distance Kernel. In *Proceedings of the 26th International Conference on Machine Learning*.

Peter Doyle and James Laurie Snell. 1984. *Random Walks and Electric Networks*. Vol. 22. Mathematical Association America, New York.

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, Philip J. Harding, Stephen G. Kobourov, Kevin Wampler, and Gary Yee. 2003. GraphAEL: Graph Animations with Evolving Layouts. In *Proceedings of the 11th International Symposium in Graph Drawing (GD), Perugia, Italy*, Vol. 2912. 98–110.

Miroslav 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.

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*.

Wolfgang Gatterbauer, Stephan Günnemann, Danai Koutra, and Christos Faloutsos. 2015. Linearized and Single-Pass Belief Propagation. *Proceedings of the VLDB Endowment* 8, 5 (2015), 581–592.

Michael Gleicher, Danielle Albers Szafir, 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.

William R. Gray, John A. Bogovic, Joshua T. Vogelstein, Bennett A. Landman, Jerry L. Prince, and R. Jacob Vogelstein. 2012. Magnetic Resonance Connectome Automated Pipeline: An Overview. *Pulse, IEEE* 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 *Proceedings of the 16th ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD), Washington, DC*. ACM, 163–172.

Tamás Horváth, Thomas Gärtner, and Stefan Wrobel. 2004. Cyclic Pattern Kernels for Predictive Graph Mining. In *Proceedings of the 10th ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD), Seattle, WA*. ACM, 158–167.

Tsuyoshi Ide and Hisashi Kashima. 2004. Eigenspace-based anomaly detection in computer systems.. In *Proceedings of the 10th ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD), Seattle, WA*. 440–449.

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 Web-Scale Graphs.. In *Proceedings of the 18th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD), Tainan, Taiwan*.

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), Anaheim, CA*.

G. Karypis and V. Kumar. 1995. METIS: Unstructured Graph Partitioning and Sparse Matrix Ordering System. *The University of Minnesota* 2 (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.

Alexander 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 *Proceedings of the 14th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD), Hyderabad, India (Lecture Notes in Computer Science)*, 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.. In *Proceedings of the 1st Conference on Email and Anti-Spam, Mountain View, CA*.

Danai Koutra, U Kang, Jilles Vreeken, and Christos Faloutsos. VoG: Summarizing and Understanding Large Graphs. In *Proceedings of the 14th SIAM International Conference on Data Mining (SDM), Philadelphia, PA*. SIAM.

Danai Koutra, U Kang, Jilles Vreeken, and Christos Faloutsos. 2015. Summarizing and Understanding Large Graphs. In *Statistical Analysis and Data Mining*. John Wiley & Sons, Inc.

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*. 245–260.

Danai Koutra, Vasileios Koutras, B. Aditya Prakash, and Christos Faloutsos. 2013. Patterns amongst Competing Task Frequencies: Super-Linearities, and the Almond-DG Model. In *Proceedings of the 17th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD)*. 201–212.

Danai Koutra, Evangelos E Papalexakis, and Christos Faloutsos. 2012. TensorSplat: Spotting Latent Anomalies in Time. In *16th Panhellenic Conference on Informatics (PCI)*. IEEE, 144–149.

Danai Koutra, Hanghang Tong, and David Lubensky. 2013a. Big-Align: Fast Bipartite Graph Alignment. In *Proceedings of the 14th IEEE International Conference on Data Mining (ICDM), Dallas, Texas*.

Danai Koutra, Joshua Vogelstein, and Christos Faloutsos. 2013b. DeltaCon: A Principled Massive-Graph Similarity Function. In *Proceedings of the 13th SIAM International Conference on Data Mining (SDM), Texas-Austin, TX*. 162–170.

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 (WWW Companion Volume)*. 195–196.

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 Mohammed 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 *International Conference on Privacy, Security, Risk and Trust (SocialCom/PASSAT)*. 193–200.

Pierre Mahé and Jean-Philippe Vert. 2009. Graph Kernels Based on Tree Patterns for Molecules. *Machine Learning* 75, 1 (April 2009), 3–35.

Ching-Hao Mao, Chung-Jung Wu, Evangelos E Papalexakis, Christos Faloutsos, and Tien-Cheu Kao. 2014. MalSpot: Multi^2 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 *IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (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 E.J. 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*. ACM, 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 Gärtner. 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).

Neil Shah, Danai Koutra, Tianmin Zou, Brian Gallagher, and Christos Faloutsos. 2015. TimeCrunch: Interpretable Dynamic Graph Summarization. In *Proceedings of the 21st ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD), Sydney, Australia*.

Nino Shervashidze and Karsten Borgwardt. 2009. Fast Subtree Kernels on Graphs. In *23rd Annual Conference on Neural Information Processing Systems (NIPS). Vancouver, British Columbia*. 1660–1668.

Nino Shervashidze, Pascal Schweitzer, Erik Jan van Leeuwen, Kurt Mehlhorn, and Karsten M. Borgwardt. 2011. Weisfeiler-Lehman Graph Kernels. *Journal of Machine Learning Research* 12 (Nov. 2011), 2539–2561.

Nino Shervashidze, S. V. N. Vishwanathan, Tobias Petri, Kurt Mehlhorn, and Karsten Borgwardt. 2009. Efficient Graphlet Kernels for Large Graph Comparison. In *Proceedings of the Twelfth International Conference on Artificial Intelligence and Statistics (AISTATS 2009)*, Vol. 5. Journal of Machine Learning Research, 488–495.

SNAP. http://snap.stanford.edu/data/index.html#web. (????).

Kumar Sricharan and Kamalika Das. 2014. Localizing Anomalous Changes in Time-evolving Graphs. In *Proceedings of the 2014 ACM International Conference on Management of Data (SIGMOD), Snowbird, UT*. ACM, 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 Conference on Information and Knowledge Management (CIKM), Maui, Hawaii*. ACM, 245–254.

Frank van Ham, Hans-Jörg Schulz, and Joan M. Dimicco. 2009. Honeycomb: Visual Analysis of Large Scale Social Networks. In *Human-Computer Interaction - INTERACT 2009*. 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*.

Heng Wang, Minh Tang, Y. Park, and C.E. Priebe. 2014. Locality Statistics for Anomaly Detection in Time Series of Graphs. *IEEE Transactions on Signal Processing* 62, 3 (Feb 2014), 703–717.

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. 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.