**Detecting Anomalous Graphs in Labeled Multi-Graph Databases**

HUNG T. NGUYEN, Princeton University, USA   
PIERRE J. LIANG, Carnegie Mellon University, USA   
LEMAN AKOGLU, Carnegie Mellon University, USA

Within a large database G containing graphs with labeled nodes and directed, multi-edges; how can we detect the anomalous graphs? Most existing work are designed for plain (unlabeled) and/or simple (unweighted) graphs. We introduce CODEtect, the irst approach that addresses the anomaly detection task for graph databases with such complex nature. To this end, it identiies a small representative set S of structural patterns (i.e., node-labeled network motifs) that losslessly compress database G as concisely as possible. Graphs that do not compress well are lagged as anomalous. CODEtect exhibits two novel building blocks: (

|  |  |  |
| --- | --- | --- |
| 2 | • | Hung T. Nguyen, et al. |

graph database. There exist related work for node-labeled graph databases [34], which however does not handle multi-edges, and as we show in the experiments (Sec. 6), cannot tackle the problem well.

Recently, general-purpose embedding/representation learning techniques achieve state-of-the-art results in graph classiication tasks [15, 16, 19, 31, 33, 37]. However, they do not tackle the anomaly detection problem per seÐthe embeddings need to be fed to an of-the-shelf vector outlier detector. Moreover, most embedding methods [15, 16, 19] produce node embeddings; how to use those for graph-level anomalies is unclear. Trivially aggregating node representations, e.g., by mean or max pooling, to obtain the entire-graph representation provides suboptimal results [31]. Graph embedding techniques [31, 37] as well as graph kernels [45, 54] (paired with a state-of-the-art detector), yield poor performance as we show through experiments (Sec. 6), possibly because embeddings capture general patterns, leaving rare structures out, which are critical for anomaly detection.

Our main contributions are summarized in the following:  
 • **Problem Formulation:** Motivated by application to business accounting, we consider the anomaly detection problem in labeled directed multi-graph (LDM) databases and propose CODEtect; (to our knowledge) the irst method to detect anomalous graphs with such complex nature (Sec. 2). CODEtect also generally applies to simpler, non-LDM settings. The main idea is to identify a few representative network motifs that are used to encode the database in a lossless fashion as succinctly as possible. CODEtect then lags those graphs that do not compress well under this encoding as anomalous (Sec. 3).

• **New Encoding & Search Algorithms:** The graph encoding problem is two-fold: how to encode and which motifs to encode with. To this end, we introduce (1) new lossless motif and graph encoding schemes (Sec. 4), and (2) eicient search algorithms for identifying key motifs with a goal to minimize the total encoding cost (Sec. 5).

• **Real-world Application:** In collaboration with industry, we apply our proposed techniques to annual transaction records from three diferent corporations, from small- to large-scale. We show the superior performance of CODEtect over existing baselines in detecting injected anomalies that mimic certain known malicious schemes in accounting. Case studies on those as well as the public Enron email database further show the efectiveness of CODEtect in spotting noteworthy instances (Sec. 6). To facilitate reproducibility, we also conirm our performance advantages on statistically similar datasets resembling our real-world databases.

**Reproducibility.** All source code as well as public-domain and synthetic data are shared at<https://bit.ly/2P0bPZQ>.

2 RELATED WORK

**Graph Anomaly Detection:** Graph anomaly detection has been studied under various settings for plain/at-tributed, static/dynamic, etc. graphs, including the most recent deep learning based approaches [1, 11, 14, 20, 38, 39, 46, 57] (See [2] and [27] for a survey.) These works focus on detecting node/edge/subgraph anomalies within a single graph, none of which applies to our setting, as we are to detect anomalous graphs (or graph-level anomalies) within a graph database.

On anomalous graph detection in graph databases, Gbad [12] has been applied to lag graphs as anomalous if they experience low compression via discovered substructures over the iterations. Further, it has been used to identify graphs that contain substructures



|  |  |  |
| --- | --- | --- |
| Detecting Anomalous Graphs in Labeled Multi-Graph Databases | • | 3 |

Table 1. Comparison with popular approaches to graph anomaly detection, in terms of distinguishing properties.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Methods vs. Properties** | Graph | Node- | Multi/ | Directed | Anomaly |
| database | labeled | Weighted | detection |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Graph Emb.** | node2vec[16], GraphSAGE[19] |  | ✔ | ✔ | ✔ |  |
| graph2vec[31],  Metagraph2vec[15], GIN[53] | ✔ | ✔ |  | ✔ |  |
| PATCHY-SAN[33],  MA-GCNN[37], Deep Graph Kernels [54] | ✔ | ✔ | ✔ | ✔ |  |
| **Graph**  **Anom. Detect.** | OddBall[1] |  |  | ✔ | ✔ | ✔ |
| FocusCO[39], AMEN[38], Dominant[11] |  | ✔ |  | ✔ | ✔ |
| GAL[57] |  |  | ✔ | ✔ | ✔ |
| CoreScope[46] |  |  |  |  | ✔ |
| FRAUDAR[20] |  |  |  | ✔ | ✔ |
| StreamSpot[28], GBAD[12] | ✔ | ✔ |  | ✔ | ✔ |
| SpotLight[14] | ✔ |  | ✔ | ✔ | ✔ |
| SnapSketch[36] | ✔ |  | ✔ |  | ✔ |
| GLocalKD[26] | ✔ | ✔ |  |  | ✔ |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| CODEtect [this paper] | ✔ | ✔ | ✔ | ✔ | ✔ |

representation approach for intrusion detection in a graph stream and showed better detection than previous works [14, 28], however, SnapSketch was originally designed to work on undirected graphs. Note also that these works [14, 28, 36] focus on graph streams, i.e., time-ordered graphs, and may not work well in our setting of unordered graph databases. We present a qualitative comparison of related work to CODEtect in Table 1. **Graph Embedding for Anomaly Detection:** Recent graph embedding methods [11, 15, 16, 19, 31, 33, 37, 57] and graph kernels [45, 54] ind a latent representation of node, subgraph, or the entire graph and have been shown to perform well on classiication and link prediction tasks. However, graph embedding approaches, like [15, 16, 19], learn a node representation, which is diicult to use directly for detecting anomalous graphs. Peng et al. [37] propose a graph convolutional neural network via motif-based attention, but, this is a supervised method and, thus, not suitable for anomaly detection. Our experimental results show that other recent graph embedding [31] and graph kernel methods [54], that produce a direct graph representation, when combined with a state-of-the-art anomaly detector have low performance and are far less accurate than CODEtect. Concurrent to our work, graph neural networks for anomalous graph detection is studied in [26, 56] that examines end-to-end graph anomaly detection. Further, [6] investigates outlier resistant architectures for graph embedding. A key challenge, in general, for deep learning based models for unsupervised anomaly detection is their sensitivity to many hyper-parameter settings (including those for regularization: such as weight decay, and drop-out rate; optimization: such as learning rate, and achitecture: such as depth, width, etc.), which are not straightforward to set in the absence of any ground-truth labels. Distinctly, our work leverages the Minimum Description Length principle and does not exhibit any hyper-parameters.

**Graph Motifs:** Network motifs have proven useful in understanding the functional units and organization of complex systems [7, 30, 51]. Motifs have also been used as features for network classiication [29], community detection [5, 55], and in graph kernels for graph comparison [45]. On the algorithmic side, several works have

ACM Trans. Knowl. Discov. Data.

|  |  |  |
| --- | --- | --- |
| 4 | • | Hung T. Nguyen, et al. |

designed fast techniques for identifying signiicant motifs [8, 9, 21, 22], where a sub-graph is regarded as a motif only if its frequency is higher than expected under a network null model.

Prior works on network motifs mainly focus on 3- or 4-node motifs in undirected unlabeled/plain graphs [4, 13, 42, 50], either using subgraph frequencies in the analysis of complex networks or most often developing fast algorithms for counting (e.g., triangles) (See [**?** ] for a recent survey). Others have also studied directed [7] and temporal motifs [24, 35]. Most relatedly, there is a recent work on node-labeled subgraphs referred to as heterogeneous network motifs [41], where again, the focus is on scalable counting. Our work difers in using heterogeneous motifs as building blocks of a graph encoding scheme, toward the goal of anomaly detection. **Data Compression via MDL-Encoding:** The MDL principle by Rissanen [40] states that the best theory to describe a data is the one that minimizes the sum of the size of the theory, and the size of the description of data using the theory. The use of MDL has a long history in itemset mining [48, 52], for transaction (tabular) data, also applied to anomaly detection [3, 47].

MDL has also been used for graph compression. Given a pre-speciied list of well-deined structures (star, clique, etc.), it is employed to ind a succinct description of a graph in those łvocabularyž terms [23]. This vocabulary is later extended for dynamic graphs [44]. A graph is also compressed hierarchically, by sequentially aggregating sets of nodes into super-nodes, where the best summary and associated corrections are found with the help of MDL [32].

There exists some work on attributed graph compression [49], but the goal is to ind super-nodes that represent a set of nodes that are homogeneous in some (user-speciied) attributes. Subdue [34] is one of the earliest work to employ MDL for substructure discovery in node-labeled graphs. The aim is to extract the łbestž substructure

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **A** | **C** | Detecting Anomalous Graphs in Labeled Multi-Graph Databases | • | 5 |
| **D** |

**B**   
 **B**

**A**

(a) (b)

Fig. 1. (a) E.g. node-labeled multi-graph where capital leters denote the node labels, and

|  |  |  |
| --- | --- | --- |
| 6 | • | Hung T. Nguyen, et al. |

**Algorithm 1** Motif Encoding

**Input:** Motif

|  |  |  |
| --- | --- | --- |
| Detecting Anomalous Graphs in Labeled Multi-Graph Databases | • | 7 |

We provide the details of how the motif usages are calculated in the next section, when we introduce graph encoding.

Next, we present how a motif

|  |  |  |
| --- | --- | --- |
| 8 | • | Hung T. Nguyen, et al. |

graph-node-IDs that correspond to or align with the motif-node-IDs. The motif-node-IDs of a motif

|  |  |  |
| --- | --- | --- |
| Detecting Anomalous Graphs in Labeled Multi-Graph Databases | • | 9 |

**Algorithm 2** Graph Encoding

**Input:** Graph

10 • Hung T. Nguyen, et al.

Let

|  |  |  |
| --- | --- | --- |
| Detecting Anomalous Graphs in Labeled Multi-Graph Databases | • | 11 |

**Algorithm 3** Greedy Algorithm for

|  |  |  |
| --- | --- | --- |
| 12 | • | Hung T. Nguyen, et al. |

**Algorithm 4** Memory-efficient Greedy

|  |  |  |
| --- | --- | --- |
| Detecting Anomalous Graphs in Labeled Multi-Graph Databases | • | 13 |

Complexity analysis: Calculating

|  |  |  |
| --- | --- | --- |
| 14 | • | Hung T. Nguyen, et al. |

**Algorithm 5** Motif Table Search

**Input:** Database G = {

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 10000 | |  | | --- | |  | | | Detecting Anomalous Graphs in Labeled Multi-Graph Databases | | | | | | | | | • | 15 |
| 10000 | |  |  |  | | --- | --- | --- | |  |  |  | | | 20000 |  | 40 | |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | |  |  | | |  |  |  |  |  |  | | | | | | | | |
| 7500 | 7500 | 15000 |
| 5000 |
| 5000 Counts  2500  0 | 10000 | 20 Counts |
| Counts |
| 2500 | 5000 | 0 |
| 0 |
| Number of nodes |
|  | 5 | 10 | 10 | 20 | 0 | 0 | 20 | 40 | 60 | |
| #nodes  (a) Distribution of the num-ber of nodes. | | | #single edges | | Label | #edges | | | | |
| (b) Distribution of the num- | | | (c) Distribution of node la- | | (d) Distribution of multiplici- | | | | | |
| ber of single edges. | | | bels. | | ties for a pair of node labels. | | | | | |

Fig. 3. Statistical summary of graph characteristics in SH database

updating

|  |  |  |
| --- | --- | --- |
| 16 | • | Hung T. Nguyen, et al. |

Table 2. Summary statistics of graph datasets used in experiments.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | #Graphs | #Node labels | #Nodes | #Multi-edges |
| SH | 38,122 | 11 | [2, 25] | [1, 782] |
| SH\_Synthetic | 38,122 | 11 | [2, 25] | [1, 782] |
| HW | 90,274 | 11 | [2, 25] | [1, 897] |
| KD | 152,105 | 10 | [2, 91] | [1, 1774] |
| Enron | 1,139 | 16 | [2, 87] | [1, 1356] |

money-laundering, where money is transferred through multiple hops rather than directly from the source to the target.

• Label injection (entry-error or malfeasance): (



|  |  |  |
| --- | --- | --- |
| Detecting Anomalous Graphs in Labeled Multi-Graph Databases | • | 17 |

Table 3. Detection performance of path anomalies in various datasets. Numbers in **bold** highlight best results in the columns and underlined numbers refer to the runner-up. In all performance measures, CODEtect achieves significantly higher results than the others and maintains a large gap to the runner-up, which is not stable but varies depending on dataset and performance measure.

(a) SH dataset

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Method** | Prec@10 Prec@100 | | Prec@1000 | AUC | AP |
| CODEtect | **1.000** | **0.920** | **0.386 0.958 0.548** | | |
| SMT | 0.100 | 0.280 | 0.352 | 0.932 | 0.413 |
| GLocalKD | 0.400 | 0.252 | 0.331 | 0.916 | 0.405 |
| Gbad | 0.200 | 0.495 | 0.356 | 0.906 | 0.373 |
| GF+iForest | 0.100 | 0.120 | 0.237 | 0.926 | 0.210 |
| G2V+iForest | 0.800 | 0.750 | 0.308 | 0.886 | 0.383 |
| DGK+iForest | 0.100 | 0.030 | 0.025 | 0.712 | 0.050 |
| Entropy | 0.100 | 0.800 | 0.219 | 0.821 | 0.347 |
| Multiedges | 0.000 | 0.040 | 0.027 | 0.643 | 0.049 |

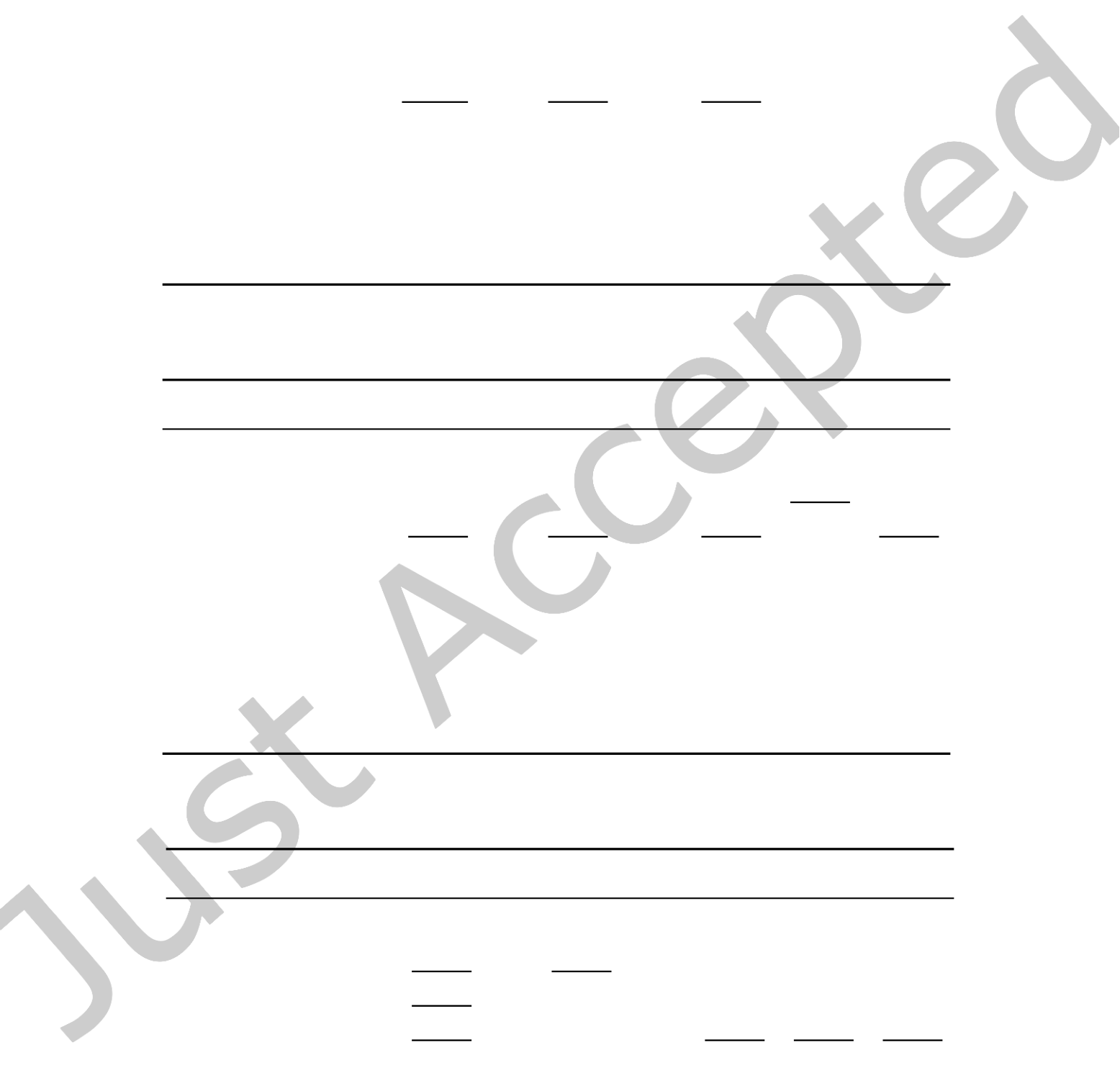
(b) HW dataset

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Method** | Prec@10 Prec@100 | | Prec@1000 | AUC | AP |
| CODEtect | **0.900** | **0.990** | **0.999 0.995 0.772** | | |
| SMT | 0.600 | 0.440 | 0.784 | 0.906 | 0.733 |
| GLocalKD | 0.000 | 0.269 | 0.531 | 0.843 | 0.563 |
| Gbad | 0.800 | 0.710 | 0.685 | 0.930 | 0.555 |
| GF+iForest | 0.400 | 0.230 | 0.497 | 0.959 | 0.429 |
| G2V+iForest | 0.000 | 0.100 | 0.819 | 0.824 | 0.380 |
| DGK+iForest | 0.300 | 0.140 | 0.023 | 0.858 | 0.097 |
| Entropy | 0.300 | 0.820 | 0.896 | 0.981 | 0.571 |
| Multiedges | 0.000 | 0.020 | 0.029 | 0.719 | 0.106 |

(c) SH\_Synthetic dataset

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Method** | Prec@10 Prec@100 | | Prec@1000 | AUC | AP |
| CODEtect | **0.600** | **0.860** | **0.412 0.982 0.664** | | |
| SMT | 0.300 | 0.440 | 0.212 | 0.911 | 0.398 |
| GLocalKD | 0.300 | 0.495 | 0.258 | 0.932 | 0.473 |
| Gbad | 0.300 | 0.523 | 0.318 | 0.937 | 0.468 |
| GF+iForest | 0.200 | 0.420 | 0.252 | 0.894 | 0.340 |
| G2V+iForest | 0.500 | 0.640 | 0.310 | 0.943 | 0.450 |
| DGK+iForest | 0.100 | 0.030 | 0.025 | 0.712 | 0.050 |
| Entropy | 0.300 | 0.540 | 0.294 | 0.921 | 0.433 |
| Multiedges | 0.000 | 0.050 | 0.036 | 0.662 | 0.102 |

ACM Trans. Knowl. Discov. Data.



|  |  |  |
| --- | --- | --- |
| 18 | • | Hung T. Nguyen, et al. |

Table 4. Detection performance of label anomalies in various datasets (**bold** and underlined numbers refer to the best and runner-up results). We observe notably higher performance of CODEtect compared to the competing baselines, where it ranks either at the top or is the runner-up across the board. In (b), − depicts cases for Gbad and G2V+iForest, which failed to complete within 5 days.

(a) HW dataset

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Method** | Prec@10 Prec@100 | | Prec@1000 | AUC | AP |
| CODEtect | **0.800** | **0.720** | **0.709** | 0.918 | 0.359 |
| SMT | 0.000 | 0.100 | 0.174 | 0.883 | 0.192 |
| GLocalKD | 0.600 | 0.640 | 0.663 | 0.902 | 0.301 |
| Gbad | **0.800** | 0.710 | 0.685 **0.920 0.555** | | |
| GF+iForest | 0.200 | 0.080 | 0.027 | 0.832 | 0.092 |
| G2V+iForest | 0.000 | 0.030 | 0.030 | 0.499 | 0.030 |
| DGK+iForest | 0.100 | 0.030 | 0.038 | 0.801 | 0.074 |
| Entropy | 0.100 | 0.030 | 0.117 | 0.623 | 0.062 |
| Multiedges | 0.000 | 0.020 | 0.032 | 0.505 | 0.030 |

(b) KD dataset

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Method** | Prec@10 Prec@100 | | Prec@1000 | AUC | AP |
| CODEtect | **0.800** | **0.940** | **0.863 0.716 0.403** | | |
| SMT | 0.200 | 0.190 | 0.122 | 0.715 | 0.082 |
| GLocalKD | 0.700 | 0.450 | 0.495 | 0.702 | 0.096 |
| Gbad | − | − | − | − | − |
| GF+iForest | 0.300 | 0.060 | 0.038 | 0.650 | 0.053 |
| G2V+iForest | − | − | − | − | − |
| DGK+iForest | 0.100 | 0.090 | 0.068 | 0.644 | 0.061 |
| Entropy | 0.400 | 0.240 | 0.130 | 0.541 | 0.040 |
| Multiedges | 0.000 | 0.040 | 0.035 | 0.498 | 0.030 |

(c) SH\_Synthetic dataset

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Method** | Prec@10 Prec@100 | | Prec@1000 | AUC | AP |
| CODEtect | **0.600** | **0.640** | **0.459 0.920 0.350** | | |
| SMT | 0.400 | 0.450 | 0.162 | 0.720 | 0.163 |
| GLocalKD | 0.400 | 0.440 | 0.321 | 0.874 | 0.289 |
| Gbad | 0.400 | 0.420 | 0.337 | 0.881 | 0.315 |
| GF+iForest | 0.200 | 0.250 | 0.140 | 0.831 | 0.127 |
| G2V+iForest | 0.000 | 0.020 | 0.021 | 0.465 | 0.020 |
| DGK+iForest | 0.200 | 0.110 | 0.063 | 0.739 | 0.125 |
| Entropy | 0.200 | 0.110 | 0.061 | 0.553 | 0.043 |
| Multiedges | 0.000 | 0.014 | 0.024 | 0.475 | 0.020 |

ACM Trans. Knowl. Discov. Data.



|  |  |  |
| --- | --- | --- |
| Detecting Anomalous Graphs in Labeled Multi-Graph Databases | • | 19 |

randomization, we also run multiple times to check the consistency of performance. Additionally, for those with hyper-parameters, we use the default settings from the corresponding publicly available source codes.

6.1.1 **Detection of path anomalies.** We report detection results on SH and HW datasets in Tables 3a and 3b (performance on KD is similar and omitted for brevity) and on SH\_Synthetic in Table 3c.

CODEtect consistently outperforms all baselines by a large margin across all performance measures in detect-ing path anomalies. More speciically, CODEtect provides 16.9% improvement over the runner-up (underlined) on average across all measures on SH, and 10.2% on HW. Note that the runner-up is not the same baseline consistently across diferent performance measures. Beneits of motif search is evident looking at the superior results over SMT. G2V+iForest produces decent performance w.r.t. most measures but is still much lower than those of CODEtect. Similar observations are present on SH\_Synthetic.

6.1.2 **Detection of label anomalies.** Tables 4a and 4b report detection results on the two larger datasets, HW and KD (performance on SH is similar and omitted for brevity), and Table 4c provides results on SH\_Synthetic. Note that Gbad and G2V+iForest failed to complete within 5 days on KD, thus their results are absent in Table 4b.

Note that KD is a relatively large-scale dataset, having both larger and more graphs than the other datasets.

In general, we observe similar performance advantage of CODEtect over the baselines for label anomalies. The exceptions are Gbad and GLocalKD which perform comparably, and appear to suit better for label anomalies, potentially because changing node labels disrupts structure more than the addition of a few short isolate paths. Gbad however does not scale to KD, and the runner-up on this dataset performs signiicantly worse. Similar observations are also seen on SH\_Synthetic dataset.

6.2 Case Studies

**Case 1 - Anomalous transaction records:** The original accounting databases we are provided with by our industry partner do not contain any ground truth labels. Nevertheless, they beg the question of whether CODEtect unearths any dubious journal entries that would raise an eyebrow from an economic bookkeeping perspective. In collaboration with their accounting experts, we analyze the top 20 cases as ranked by CODEtect. Due to space limit, we elaborate on one case study from each dataset/corporation as follows.

In SH, we detect a graph with a large encoding length yet relatively few (27) multi-edges, as shown in Fig. 5, consisting of several small disconnected components. In accounting terms the transaction is extremely complicated, likely the result of a (quite rare) łbusiness restructuringž event. In this single journal entry there exist many independent simple entries, involving only one or two operating-expense (OE) accounts, while other edges arise from compound entries (involving more than three accounts). This event involves reversals (back to prepaid expenses) as well as re-classiication of previously booked expenses. The fact that all these bookings are recorded within a single entry leaves room for manipulation of economic performance and mis-reporting via re-classiication, which deserves an audit for careful re-examination.

In Fig. 4 (left) we show a motif with sole usage of 1 in the dataset, which is used to cover an anomalous graph (right) in HW. Edge NGL (non-operating gains&losses) to C (cash) depicts an unrealized foreign exchange gain and is quite unusual from an economic bookkeeping perspective. This is because, by deinition, unrealized gains and losses do not involve cash. Therefore, proper booking of the creation or relinquishment of such gains or losses should not involve cash accounts. Another peculiarity is the three separate disconnected components, each of which represents very distinct economic transactions: one on a bank charge related to a security deposit, one on health-care and travel-related foreign-currency business expense (these two are short-term activities), and a third one on some on-going construction (long-term in nature). It is questionable why these diverse transactions are grouped into a single journal. Finally, the on-going construction portion involves reclassifying a long-term asset into a suspense account, which requires follow-up attention and inal resolution.

ACM Trans. Knowl. Discov. Data.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 20 | • | Hung T. Nguyen, et al. | NGL | NGL | LOL | NGL | NGL | 1 | SOA | SOA |
| C | 1 | 1 | 1 |
| C | OE | LOA |
| 1 |
| LOL |

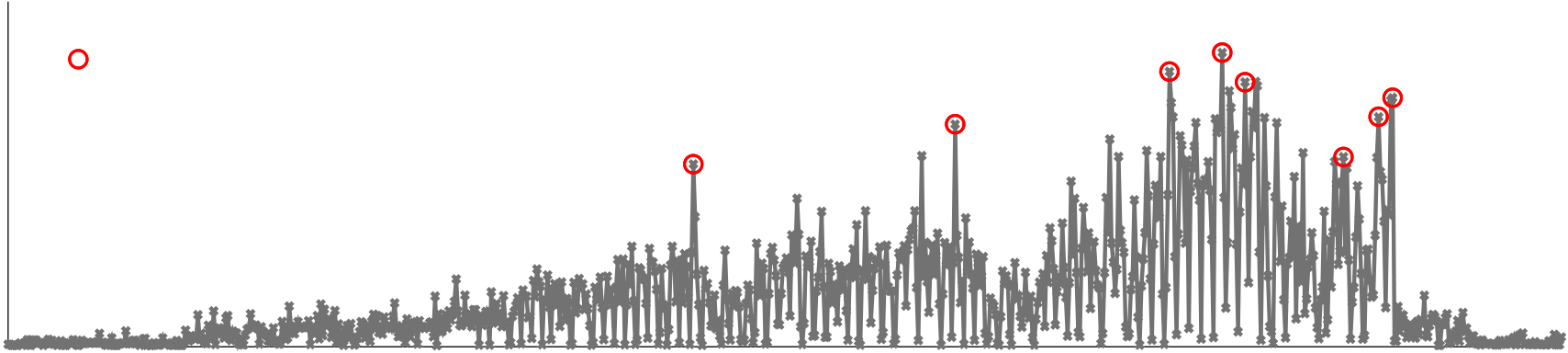
Fig. 4. (let) A rare motif, (right) Anomalous graph in HW.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| SOL | OE | OE | 1 | 1 | OE | 1 | 1 | OE | 1 | OE | | OE | 1 | 1 | OE | OE | 1 | OE | 1 | OE | 1 | OE | 1 | OE | 1 | OE | 1 |
| 1 | 1 | 1 | 1 | | 1 | 1 | 1 | 1 |
| OE | OE | SOA | | OE | OE | OE | OE |
| OE | 1 | 3 | 1 | 1 |
| OE | | OE | NGL |

Fig. 5. Anomalous graph in SH.

|  |  |  |  |
| --- | --- | --- | --- |
| Finally, the anomalous journal entry from KD involves the motif shown | LOA | LOA | 1 |
| in Fig. 6 (left) where the corresponding graph is the exact motif with | 1 |
| multiplicity 1 shown on the (right). This motif has sole usage of 1 in | LOA |
| LOA |
| the dataset and is odd from an accounting perspective. Economically, it |
| represents giving up an existing machine, which is a long-term operating | 1 |
| asset (LOA), in order to reduce a payable or an outstanding short-term | SOL |
| operating liability (SOL) owed to a vendor. Typically one would sell the | SOL |

machine and get cash to payof the vendor with some gains or losses. We also note that the



|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Code length | Major Events | E1 | Detecting Anomalous Graphs in Labeled Multi-Graph Databases | | | • | 21 |
| E2 | E3 E4 E5 | E7E8 |
| E6 | | |

Daily graph index

|  |  |  |
| --- | --- | --- |
| E1 | Dec 13, 2000 | Enron announces that president and COO Jefrey Skilling will take over as chief |

executive in February.

|  |  |  |
| --- | --- | --- |
| E2 | May 23, 2001 | Enron completes its 1,000,000-th transaction via Enron Online. |
| E3 | Sep 26, 2001 | Employee Meeting. Kenneth Lay tells employees: Enron stock is an łincredible bar- |

gainž and łThird quarter is looking great.ž

|  |  |  |
| --- | --- | --- |
| E4 | Oct 24-29, 2001 | Enron sacks Andrew Fastow. In vain Lay calls chairman of the Fed and the Treasury |

and Commerce secretaries to solicit help.

|  |  |  |
| --- | --- | --- |
| E5 | Nov 9, 2001 | Dynegy agrees to buy Enron for about $9 billion in stock and cash. |
| E6 | Jan 10, 2002 | DOJ conirms criminal investigation begun. Arthur Andersen announces employees |

in Houston Division had destroyed documents.

|  |  |  |
| --- | --- | --- |
| E7 | Jan 30, 2002 | Enron names Stephen F. Cooper new CEO. |
| E8 | Feb 07, 2002 | A. Fastow invokes the Fifth Amendment before Congress; J. Skilling testiies he knew |

of no wrongdoing at Enron when he resigned.

Fig. 7. Code lengths of Enron’s daily email exchange graphs. Large values coincide with key events of the financial scandal.

also noteworthy that the anomaly scores follow an increasing trend over days, capturing the escalation of events up to key personnel testifying in front of Congressional committees.

6.3 Scalability

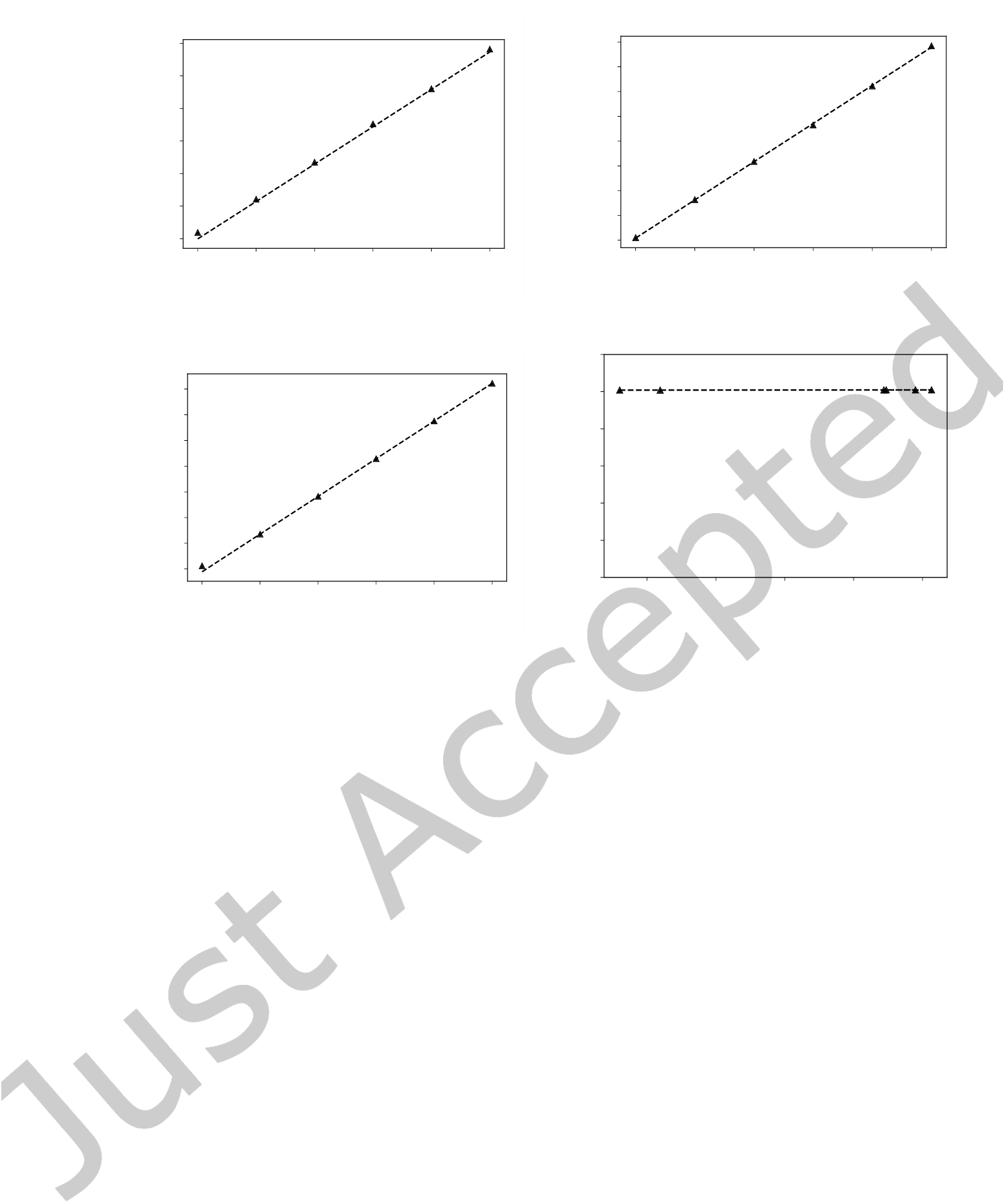
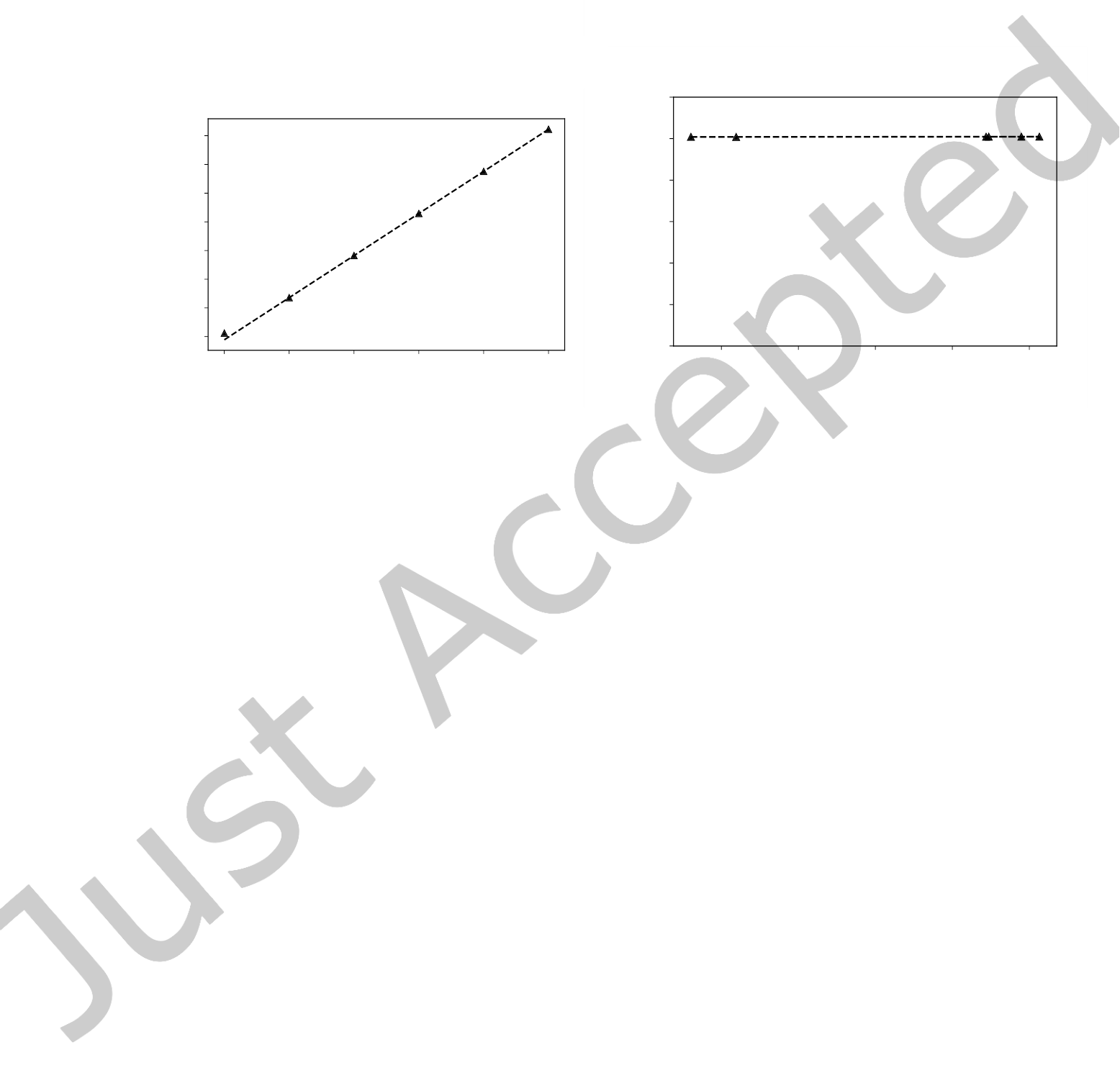
To showcase the scalability of CODEtect, in regard to running time and memory consumption, we randomly selected subsets of graphs in KD database with diferent sizes, i.e., {40, 45, 50, 55, 60, 65, 70, 75, 80, 85, 90, 95, 100} ×103. We re-sample each subset of graphs three times and report the averaged result for each setting. A summary of the results is presented in Figure 8. We observe a linear scaling of CODEtect with increasing size of input graphs as measured in number of multi-edges with respect to both time and memory usage.

7 CONCLUSION

We introduced CODEtect, (to our knowledge) the irst graph-level anomaly detection method for node-labeled multi-graph databases; which appear in numerous real-world settings such as social networks and inancial transactions, to name a few. The main idea is to identify key network motifs that encode the database concisely and

employ compression length as the anomaly score. To this end, we presented (1) novel lossless encoding schemes and (2) eicient search algorithms. Experiments on transaction databases from three diferent corporations quantitatively showed that CODEtect signiicantly outperforms the prior and more recent GNN based baselines across datasets and performance metrics. Case studies, including the Enron database, presented qualitative evidence to CODEtect’s efectiveness in spotting instances that are noteworthy of auditing and re-examination.

ACM Trans. Knowl. Discov. Data.



|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 22 | • | Hung T. Nguyen, et al. | | | | Memory Usage of CODEtect (training) | | | |
| Running time of CODEtect (training) | | | |
| 600 | | | | 2500 | | | |
| 500 | | | | 2250 | | | |
| 2000 | | | |
| 400 | | | | 1750 | | | |
| 300 | | | | 1500 | | | |
| 200 | | | | 1250 | | | |
| 1000 | | | Linear Scaling |
| 100 | Linear Scaling | | |
| 750 | | | |
| 0 | | | | 500 | | | |
| Running time (minutes) | 105 | 2×1053×1054×1055×1056×105 | | 105 | | 2×1053×1054×1055×1056×105 | |
| Number of multi-edges | | | | Memory usage (Megabytes) | Number of multi-edges | | |
| Running time of CODEtect (scoring) | | | | Memory Usage of CODEtect (scoring) 42 | | | |
| 2.25 | | | | 40 | | | |
| 2.00 | | | | 38 | | | |
| 1.75 | | | |
| 36 | | | |
| 1.50 | | | |
| 1.25 | | | | 34 | | | |
| 1.00 | | | | 32 | | | |
| 0.75 | | | Linear Scaling |
| 0.50 | | | | 30 | 1.9×1012×1012.1×1012.2×1012.3×101 | | |
| Running time (seconds) | 105 | 2×1053×1054×1055×1056×105 | |
| Number of multi-edges | | | | Average memory usage (MB) | Average number of multi-edges | | |

Fig. 8. (let) Time, (right) Memory consumption (training refers to running the full motif table search, while scoring assumes the motif table has been learned and only executes graph encoding for anomaly scoring.) Note that CODEtect training and scoring both scale linearly in the number of multi-edges w.r.t. both time and memory usage.

ACKNOWLEDGMENTS

This work is sponsored by NSF CAREER 1452425 and the PwC Risk and Regulatory Services Innovation Center at Carnegie Mellon University. Any conclusions expressed in this material are those of the author and do not necessarily relect the views, expressed or implied, of the funding parties.

REFERENCES

[1] Leman Akoglu, Mary McGlohon, and Christos Faloutsos. 2010. oddball: Spotting Anomalies in Weighted Graphs.. In PAKDD, Vol. 6119. Springer, 410ś421.

[2] Leman Akoglu, Hanghang Tong, and Danai Koutra. 2015. Graph based anomaly detection and description: a survey. Data Min. Knowl. Discov. 29, 3 (2015), 626ś688.

[3] Leman Akoglu, Hanghang Tong, Jilles Vreeken, and Christos Faloutsos. 2012. Fast and reliable anomaly detection in categorical data.. In CIKM. 415ś424.

[4] Mohammad Al Hasan and Vachik S. Dave. 2018. Triangle counting in large networks: a review. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery 8, 2 (2018), e1226.

[5] A. Arenas, A. Fernández, S. Fortunato, and S. Gómez. 2008. Motif-based communities in complex networks. Journal of Physics A 41, 22 (2008), 224001.

[6] Sambaran Bandyopadhyay, Saley Vishal Vivek, and MN Murty. 2020. Outlier resistant unsupervised deep architectures for attributed network embedding. In Proceedings of the 13th International Conference on Web Search and Data Mining. 25ś33.

[7] Austin R. Benson, David F. Gleich, and Jure Leskovec. 2016. Higher-order organization of complex networks. Science 353, 6295 (2016), 163ś166.

|  |  |  |
| --- | --- | --- |
| [8] Peter Bloem and Steven de Rooij. 2017. | Large-Scale Network Motif Learning with Compression. | CoRR abs/1701.02026 (2017). |

arXiv[:1701.02026](https://arxiv.org/abs/1701.02026)   
[9] Peter Bloem and Steven de Rooij. 2020. Large-scale network motif analysis using compression. Data Mining and Knowledge Discovery 34, 5 (2020), 1421ś1453.

ACM Trans. Knowl. Discov. Data.



|  |  |  |
| --- | --- | --- |
| Detecting Anomalous Graphs in Labeled Multi-Graph Databases | • | 23 |

[10] Diane J. Cook and Lawrence B. Holder. 1994. Substructure Discovery Using Minimum Description Length and Background Knowledge. JAIR 1 (1994), 231ś255.

[11] Kaize Ding, Jundong Li, Rohit Bhanushali, and Huan Liu. 2019. Deep anomaly detection on attributed networks. In Proceedings of the 2019 SIAM International Conference on Data Mining. SIAM, 594ś602.

[12] William Eberle and Lawrence B. Holder. 2007. Anomaly detection in data represented as graphs. Intell. Data Anal. 11, 6 (2007), 663ś689. [13] Ethan R. Elenberg, Karthikeyan Shanmugam, Michael Borokhovich, and Alexandros G. Dimakis. 2015. Beyond Triangles: A Distributed Framework for Estimating 3-proiles of Large Graphs.. In KDD, Longbing Cao, Chengqi Zhang, Thorsten Joachims, Geofrey I. Webb, Dragos D. Margineantu, and Graham Williams (Eds.). ACM, 229ś238.

[14] Dhivya Eswaran, Christos Faloutsos, Sudipto Guha, and Nina Mishra. 2018. SpotLight: Detecting Anomalies in Streaming Graphs.. In KDD. ACM, 1378ś1386.

[15] Yujie Fan, Shifu Hou, Yiming Zhang, Yanfang Ye, and Melih Abdulhayoglu. 2018. Gotcha-Sly Malware! Scorpion A Metagraph2vec Based Malware Detection System. In KDD. 253ś262.

[16] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In KDD. 855ś864.

[17] Peter D Grünwald. 2007. The minimum description length principle. MIT press.

[18] Magnús M Halldórsson and Jaikumar Radhakrishnan. 1997. Greed is good: Approximating independent sets in sparse and bounded-degree graphs. Algorithmica 18, 1 (1997), 145ś163.

[19] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In NeurIPS. 1024ś1034. [20] Bryan Hooi, Kijung Shin, Hyun Ah Song, Alex Beutel, Neil Shah, and Christos Faloutsos. 2017. Graph-based fraud detection in the face of camoulage. TKDD 11, 4 (2017), 1ś26.

[21] Zahra Kashani, Hayedeh Ahrabian, Elahe Elahi, Abbas Nowzari-Dalini, Elnaz Ansari, Sahar Asadi, Shahin Mohammadi, Falk Schreiber, and Ali Masoudi-Nejad. 2009. Kavosh: a new algorithm for inding network motifs. BMC Bioinformatics 10, 1 (2009), 318.

[22] Nadav Kashtan, Shalev Itzkovitz, Ron Milo, and Uri Alon. 2004. Eicient sampling algorithm for estimating subgraph concentrations and detecting network motifs. Bioinformatics 20, 11 (2004), 1746ś1758.

[23] Danai Koutra, U. Kang, Jilles Vreeken, and Christos Faloutsos. 2014. VOG: Summarizing and Understanding Large Graphs.. In SDM. SIAM, 91ś99.

[24] Lauri Kovanen, Márton Karsai, Kimmo Kaski, János Kertész, and Jari Saramäki. 2011. Temporal motifs in time-dependent networks. Journal of Statistical Mechanics: Theory and Experiment 2011, 11 (2011), P11005.

[25] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. 2012. Isolation-based anomaly detection. TKDD 6, 1 (2012), 3.

[26] Rongrong Ma, Guansong Pang, Ling Chen, and Anton van den Hengel. 2022. Deep Graph-level Anomaly Detection by Glocal Knowledge Distillation. In Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining. 704ś714.

[27] Xiaoxiao Ma, Jia Wu, Shan Xue, Jian Yang, Chuan Zhou, Quan Z Sheng, Hui Xiong, and Leman Akoglu. 2021. A comprehensive survey on graph anomaly detection with deep learning. IEEE Transactions on Knowledge and Data Engineering (2021).

[28] Emaad A. Manzoor, Sadegh M. Milajerdi, and Leman Akoglu. 2016. Fast Memory-eicient Anomaly Detection in Streaming Heterogeneous Graphs.. In KDD. ACM, 1035ś1044.

[29] Ron Milo, Shalev Itzkovitz, Nadav Kashtan, Reuven Levitt, Shai Shen-Orr, Inbal Ayzenshtat, Michal Shefer, and Uri Alon. 2004. Superfamilies of evolved and designed networks. Science 303, 5663 (2004), 1538ś1542.

[30] Ron Milo, Shai Shen-Orr, Shalev Itzkovitz, Nadav Kashtan, Dmitri Chklovskii, and Uri Alon. 2002. Network Motifs: Simple Building Blocks of Complex Networks. Science 298, 5594 (2002), 824ś827.

[31] Annamalai Narayanan, Mahinthan Chandramohan, Rajasekar Venkatesan, Lihui Chen, Yang Liu, and Shantanu Jaiswal. 2017. graph2vec: Learning distributed representations of graphs. arXiv preprint arXiv:1707.05005 (2017).

[32] Saket Navlakha, Rajeev Rastogi, and Nisheeth Shrivastava. 2008. Graph summarization with bounded error.. In SIGMOD. ACM, 419ś432. [33] Mathias Niepert, Mohamed Ahmed, and Konstantin Kutzkov. 2016. Learning convolutional neural networks for graphs. In ICML. 2014ś2023.

[34] Caleb C. Noble and Diane J. Cook. 2003. Graph-based anomaly detection.. In KDD. ACM, 631ś636.

[35] Ashwin Paranjape, Austin R. Benson, and Jure Leskovec. 2017. Motifs in Temporal Networks. In WSDM. ACM, 601ś610.

[36] Ramesh Paudel and William Eberle. 2020. SNAPSKETCH: Graph Representation Approach for Intrusion Detection in a Streaming Graph. In MLG 2020: 16th International Workshop on Mining and Learning with Graphs. ACM.

[37] Hao Peng, Jianxin Li, Qiran Gong, Senzhang Wang, Yuanxing Ning, and Philip S Yu. 2018. Graph convolutional neural networks via motif-based attention. arXiv preprint arXiv:1811.08270 (2018).

[38] Bryan Perozzi and Leman Akoglu. 2016. Scalable anomaly ranking of attributed neighborhoods. In SDM. SIAM, 207ś215. [39] Bryan Perozzi, Leman Akoglu, Patricia Iglesias Sánchez, and Emmanuel Müller. 2014. Focused clustering and outlier detection in large attributed graphs. In KDD. 1346ś1355.

[40] Jorma Rissanen. 1978. Modeling by shortest data description. Automatica 14, 5 (1978), 465ś471.

[41] Ryan A. Rossi, Nesreen K. Ahmed, Aldo G. Carranza, David Arbour, Anup Rao, Sungchul Kim, and Eunyee Koh. 2019. Heterogeneous Network Motifs. CoRR abs/1901.10026 (2019). arXiv[:1901.10026](https://arxiv.org/abs/1901.10026)

ACM Trans. Knowl. Discov. Data.



|  |  |  |
| --- | --- | --- |
| 24 | • | Hung T. Nguyen, et al. |

[42] Seyed-Vahid Sanei-Mehri, Ahmet Erdem Sariyüce, and Srikanta Tirthapura. 2018. Butterly Counting in Bipartite Networks.. In KDD, Yike Guo and Faisal Farooq (Eds.). ACM, 2150ś2159.

[43] ]SeTi19 C. Seshadhri and Srikanta Tirthapura. [n. d.]. Scalable Subgraph Counting: The Methods Behind The Madness: WWW 2019 Tutorial. In WWW.

[44] Neil Shah, Danai Koutra, Tianmin Zou, Brian Gallagher, and Christos Faloutsos. 2015. TimeCrunch: Interpretable Dynamic Graph Summarization.. In KDD. ACM, 1055ś1064.

[45] Nino Shervashidze, S. V. N. Vishwanathan, Tobias Petri, Kurt Mehlhorn, and Karsten M. Borgwardt. 2009. Eicient graphlet kernels for large graph comparison. In AISTATS. 488ś495.

[46] Kijung Shin, Tina Eliassi-Rad, and Christos Faloutsos. 2016. Corescope: Graph mining using k-core analysisÐpatterns, anomalies and algorithms. In ICDM. IEEE, 469ś478.

[47] Koen Smets and Jilles Vreeken. 2011. The Odd One Out: Identifying and Characterising Anomalies.. In SDM. SIAM / Omnipress, 804ś815.

[48] Nikolaj Tatti and Jilles Vreeken. 2008. Finding Good Itemsets by Packing Data.. In ICDM. IEEE Computer Society, 588ś597. [49] Yuanyuan Tian, Richard A. Hankins, and Jignesh M. Patel. 2008. Eicient aggregation for graph summarization.. In SIGMOD. ACM, 567ś580.

[50] Johan Ugander, Lars Backstrom, and Jon M. Kleinberg. 2013. Subgraph frequencies: mapping the empirical and extremal geography of large graph collections.. In WWW, Daniel Schwabe, Virgílio A. F. Almeida, Hartmut Glaser, Ricardo A. Baeza-Yates, and Sue B. Moon (Eds.). ACM, 1307ś1318.

[51] A. Vázquez, R. Dobrin, D. Sergi, J.-P. Eckmann, Z. N. Oltvai, and A.-L. Barabási. 2004. The topological relationship between the large-scale attributes and local interaction patterns of complex networks. PNAS 101, 52 (2004), 17940ś17945.

[52] Jilles Vreeken, Matthijs van Leeuwen, and Arno Siebes. 2011. Krimp: mining itemsets that compress. Data Min. Knowl. Discov. 23, 1 (2011), 169ś214.

[53] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2019. How Powerful are Graph Neural Networks?. In ICLR.

[54] Pinar Yanardag and SVN Vishwanathan. 2015. Deep graph kernels. In KDD. 1365ś1374.

[55] Hao Yin, Austin R. Benson, and Jure Leskovec. 2018. Higher-order clustering in networks. Phys. Rev. E 97 (2018), 052306. [56] Lingxiao Zhao and Leman Akoglu. 2021. On using classiication datasets to evaluate graph outlier detection: Peculiar observations and new insights. Big Data (2021).

[57] Tong Zhao, Chuchen Deng, Kaifeng Yu, Tianwen Jiang, Daheng Wang, and Meng Jiang. 2020. Error-Bounded Graph Anomaly Loss for GNNs. In Proceedings of the 29th ACM International Conference on Information & Knowledge Management. 1873ś1882.

ACM Trans. Knowl. Discov. Data.