VII. Supplementary Materials

In this section, we provide the information we omitted in the paper due to the space limit.

A. Proof of Time Complexity

Proof. In the worst case, n frequent substructures are detected by $gSpan$ and all can be used for compression with no conflict, then GAWD at most will iterate n times. Moreover, $O(I|E|\log|V|)$ is the time complexity of $gSpan$, where I denotes the number of graphs in graph database, $|E|$ denotes the average edge number of graphs, and $|V|$ denotes the average node number. The Dichotomous Search is also efficient, taking $O(|E_{j,(u,v)}|\log_2 R)$, where $E_{j,(u,v)}$ denotes all the edges in the instances of substructure P_j, and R denotes the numeric search range of weights. For compression, it takes $O(I|E|)$ to redirect edges for each graph. The complexity of GAWD is $O(n(|E_{j,(u,v)}|\log_2 R + I|E| + I|E|\log|V|))$. Empirically, $|E_{j,(u,v)}|$ and $\log_2 R$ are small constant values which are negligible. Therefore, the complexity is $O(nI|E|\log|V|)$. ■

B. Random Transaction Graph Database Generation

1) Algorithm: Algorithm 3 illustrates the procedure of generation. In line 2, we extract the statistics from the given transaction graph database G, where D_{in} and D_{out} denote the in- and out-degree sequences for each graph respectively, P_t denotes the probability of node types in the database, P_e denotes the probability of edge types, and P_w denotes the probability of weights given an edge type. We then create the random graph by in- and out-degree sequences of each graph with Directed Havel Hakimi Graph algorithm [18] in line 4. We will skip the sequence set that the algorithm cannot give a connected graph after a number of trials. In line 5, we randomly initialize the first node and assign the label by P_t. We then greedily assign labels to other nodes by P_e in line 6-11. To those unlabeled nodes and unweighted edges, we finish the assignment after the greedy one.

Data: A transaction graph database G
Result: A random transaction graph database G^*
1 $G^* \leftarrow \emptyset$;
2 Extract $D_{in}, D_{out}, P_t, P_e$ and P_w from G;
3 **for** $d_{in} \in D_{in}, d_{out} \in D_{out}$ **do**
4 Create a random graph g by [18] with d_{in} and d_{out};
5 Randomly select a node n_i and assign a label by P_t;
6 **while** n_i *exists any neighbour without label* **do**
7 Randomly select n_j from neighbours without label;
8 Assign a label to n_j by P_e;
9 Assign a weight to edge (n_i, n_j) by P_w;
10 $n_i \leftarrow n_j$;
11 **end**
12 Assign labels to those unlabeled nodes by P_t;
13 Assign weights to those unweighted edges by P_w;
14 $G^* \leftarrow G^* \cup g$;
15 **end**
16 Return G^*;

Algorithm 3: Random Transaction Graph Database Generator

2) Experimental Results: Here we examine the similarity between the real-world and random transaction graph database. Since we use exactly the same degree sequences to generate the graphs, the distribution of node and edge number are the same as well. Figure 3 illustrates the distributions of specific structures in the transaction graphs, where the random database has very similar distributions as the real-world one. Number of triangles is a common statistical measure for the graphs, and bidirectional edges represent an essential meaning in transaction graphs, denoting the mutual money transfers. In Figure 4, we show that the distributions node type, edge weight and edge type are extremely similar, except for some of the ones with small number of occurrences. Figure 5 depicts the edge weight distribution for each edge type, where we can find that they are also very similar. In summary, experimental results demonstrate that our random transaction graph database statistically follows the real-world one.

C. Detailed Descriptions of Datasets

Here we provide the complete information of datasets used in the experiments. One common setting among all the datasets is that we treat edge multiplicities as weights. We make three datasets with node labels and edge weights (both original and injected) publicly available on https://github.com/mengchillee/GAWD/tree/master/data, namely UCI Message, Enron Email and Random Accounting datasets. The detailed description of all datasets are shown as follows:

- **UCI Message Dataset [16]:** It recorded the communications between students at the University of California, Irvine, where nodes denote students and edges denote sent messages. To assign node labels, we use role2vec [19] to embed nodes by capturing their role information in the complete graph. We then use the 10 groups clustered by Agglomerative Clustering with the embeddings as the node labels. The data is split into hours to form a graph database.
- **Enron Email Dataset [17]:** It contains the email passing between colleagues in Enron Company during 2000 to 2002. We assign the job positions to each employee as node labels. The data is split into days to form a graph database.
- **Accounting Dataset:** It is from an anonymous institution, containing accounts (nodes) and transactions (edges) that precisely reflect the money flow between company accounts. Each graph captures a set of transactions within a unique expense report.
- **Random Accounting Dataset:** Because of the privacy issue, we generate the random transaction graph database by the algorithm described in Section VII-B. The generated dataset statistically follows the real-world accounting dataset.

D. Effectiveness with Recall and Run Time

To evaluate the performance of anomaly detection, we use precision at k, recall at k, Area Under Curve (AUC) and Average Precision (AP) as our evaluation metrics. The

choices of k are dependent on the database size since k cannot exceed the number of injected graphs. We show the detailed experimental results of UCI Message, Enron Email, Accounting and Random Accounting datasets in Table IV, Table V, Table VI and Table VII, respectively. The results demonstrate that GAWD achieves the best performance as well as the competitive running time.

TABLE IV: **Performance on detecting anomalies on UCI Message Dataset**

Method	Pre@45	Pre@90	Rec@45	Rec@90	AUC	AP	Time
node2vec	6.7	5.6	3	5.1	47.6	3.1	58.6s
graph2vec	2.2	1.1	1.0	1.0	48.7	3.1	2.9s
Noble *et al.*	0.0	0.0	0.0	0.0	47.6	3.1	43988s
Subdue-W	**100.0**	60.0	**45.5**	54.5	**94.8**	68.6	32621s
GAWD	**100.0**	**92.2**	**45.5**	**83.8**	93.8	**90.3**	760s

TABLE V: **Performance on detecting anomalies on Enron Email Dataset**

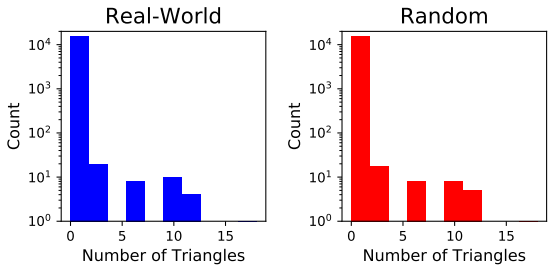
Method	Pre@10	Pre@20	Rec@10	Rec@20	AUC	AP	Time
node2vec	10.0	5.0	4.0	4.0	58.7	4.6	24.6s
graph2vec	10.0	5.0	4.0	4.0	59.0	6.1	1.1s
Noble *et al.*	0.0	0.0	0.0	0.0	51.7	3.2	7768s
Subdue-W	10.0	5.0	4.0	4.0	48.2	4.9	8443s
GAWD	**90.0**	**85.0**	**36.0**	**68.0**	**82.1**	**73.8**	207s

TABLE VI: **Performance on detecting anomalies on Accounting Dataset**

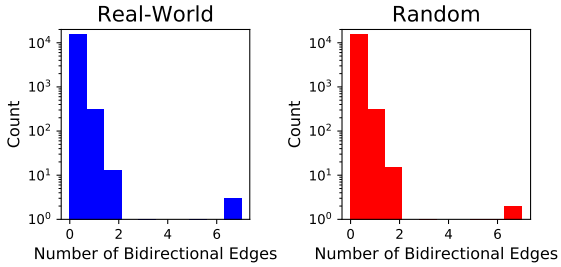
Method	Pre@200	Pre@400	Rec@200	Rec@400	AUC	AP	Time
node2vec	5.0	3.0	2.1	2.5	47.0	30.	31.3s
graph2vec	3.5	4.5	1.5	3.8	54.0	3.7	12.7s
Noble *et al.*	3.0	3.1	1.2	2.6	50.6	3.1	460s
Subdue-W	82.0	74.0	34.2	61.7	76.0	59.8	477s
GAWD	**100.0**	**81.5**	**41.7**	**67.9**	**88.0**	**71.0**	75s

TABLE VII: **Performance on detecting anomalies on Random Accounting Dataset**

Method	Pre@200	Pre@400	Rec@200	Rec@400	AUC	AP	Time
node2vec	2.5	3.3	1.0	0.8	48.3	2.8	25.9s
graph2vec	0.0	2.0	1.7	3.1	49.6	3	14s
Noble *et al.*	3.0	3.0	1.3	2.5	50.0	3	1426s
Subdue-W	63.5	35	26.6	29.3	71.5	36.8	6584s
GAWD	**100.0**	**89.0**	**41.8**	**74.5**	**95.3**	**90.2**	176s

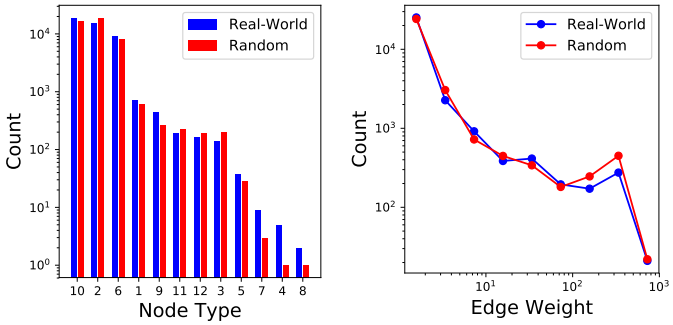


(a) Triangle Number

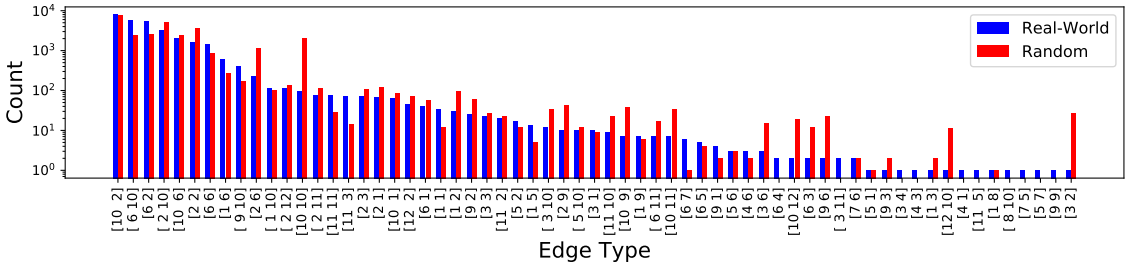


(b) Bidirectional Edge Number

Fig. 3: Comparison on distributions of triangle number and bidirectional edge number

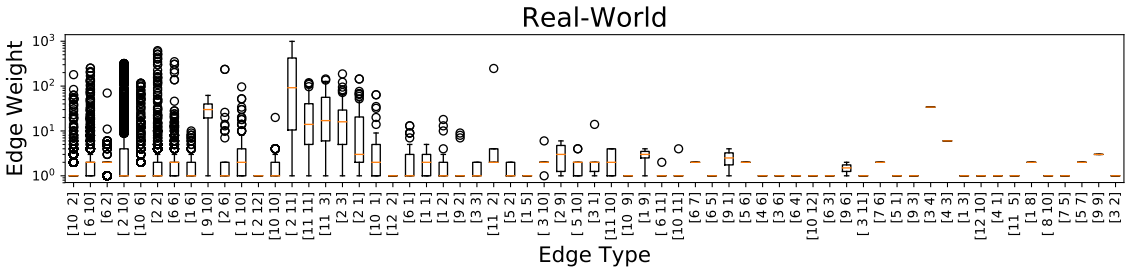


(a) Node Type (b) Edge Weight

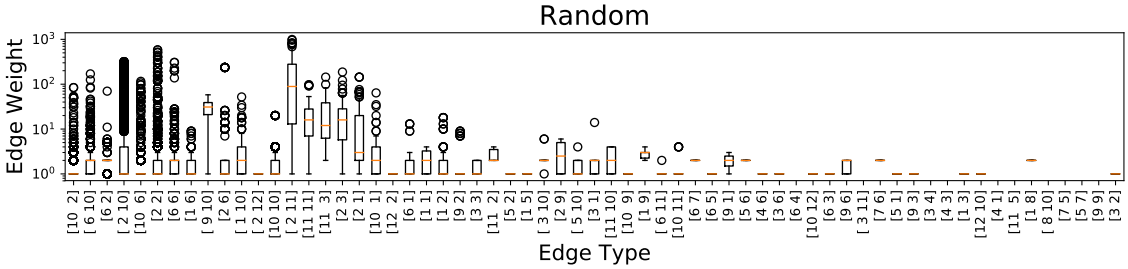


(c) Edge Type

Fig. 4: Comparison on distributions of node type, edge weight and edge type



(a) Real-World Database



(b) Random Database

Fig. 5: Box plot of edge type versus edge weight