G2A2: Graph Generator with Attributes and Anomalies

Saikat Dey*
Virginia Tech
Blacksburg, Virginia, USA
dsaikat@vt.edu

Sonal Jha*
Virginia Tech
Blacksburg, Virginia, USA
sonalj@vt.edu

Wu-chun Feng
Virginia Tech
Blacksburg, Virginia, USA
wfeng@vt.edu

ABSTRACT

Many data-mining applications use dynamic attributed graphs to represent relational information; but due to security and privacy concerns, there is a dearth of publicly available datasets that can be represented as dynamic attributed graphs. Even when such datasets are available, they do *not* have ground truth that can be useful for classification problems, e.g., anomaly detection. Thus, researchers commonly generate synthetic graphs using either statistical or deep generative (DG) methods. However, neither approach produces ground truth. Statistical methods struggle to replicate intricate patterns found in real-world dynamic attributed graphs, while DG methods require a significant number of graphs for training.

To address these shortcomings, we present G2A2, an automated graph generator with attributes and anomalies, which encompasses (1) probabilistic models to generate a dynamic bipartite graph, representing realistic time-evolving connections between two independent sets of entities, (2) realistic injection of anomalies for ground truth using a novel algorithm that captures the general properties of graph anomalies across domains, and (3) generative adversarial network (GAN) model to produce realistic attributes, learned from an existing real-world dataset. We also show that G2A2 is scalable and can generate a graph with a million edges in under a minute of computing time. Using the maximum mean discrepancy (MMD) metric to evaluate the realism of a G2A2-generated graph against three real-world graphs, G2A2 outperforms Kronecker graph generation by reducing the MMD distance by up to six-fold (6×).

CCS CONCEPTS

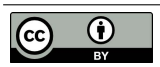
• **Networks** → **Topology analysis and generation**; • **Security and privacy** → *Intrusion/anomaly detection and malware mitigation.*

KEYWORDS

Graph generation, dynamic graphs, bipartite graphs, ground truth

ACM Reference Format:
Saikat Dey, Sonal Jha, and Wu-chun Feng. 2024. G2A2: Graph Generator with Attributes and Anomalies. In *21st ACM International Conference on Computing Frontiers (CF '24), May 7–9, 2024, Ischia, Italy.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3649153.3649206

*Both authors contributed equally to the paper



CF '24, May 7–9, 2024, Ischia, Italy

ACM ISBN 979-8-4007-0597-7/24/05
https://doi.org/10.1145/3649153.3649206

1 INTRODUCTION

Dynamic attributed graphs represent relational information in many data-mining applications, including fraud detection in commerce, intrusion detection in networking, and recommendation systems in social media. However, there is a *scarcity* of publicly available data that can be represented as dynamic attributed graphs due to security, privacy, and obfuscation. Even rarer are datasets with *ground truth* that are useful for classification problems, e.g., anomaly detection [7]. Consequently, researchers generate synthetic graphs to fill this void by using either deep generative (DG) models or statistical models, as illustrated in Figure 1.

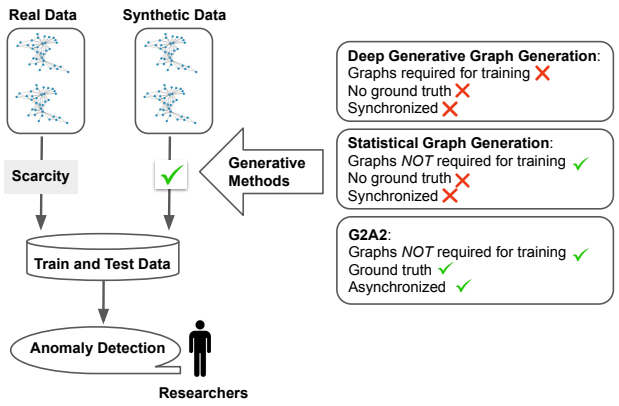


Figure 1: Data requirement for anomaly detection

DG models for graph generation require a significant number of dynamic attributed graphs for training and do *not* produce ground truth. On the other hand, while statistical models for graph generation do *not* require graphs for training, they also do *not* produce ground truth. Scalability is yet another aspect where both DG and statistical graph generators fall short. Most DG models and statistical models are synchronized in nature and not scalable. For example, DG models face memory constraints that limit scalability [37] while sophisticated statistical models, such as the multiplicative attribute graph (MAG) [15] model, can take hours to generate a graph with millions of nodes and edges. The DG and statistical graph generators have further limitations, as articulated below.

DG methodologies can generate realistic graphs with (D2G2 [35]) or without attributes (TagGen [36]); however, none of these models can generate ground truth or special graphs, like bipartite graphs that represent connections between two independent sets of entities (e.g., users and items) commonly found in domains like social media.

While statistical graph generation models, e.g., Erdős-Rényi (E-R) [10] and Kronecker [18], are widely used to generate both static and dynamic graphs, they are overly simplistic and fail to capture

the intricate patterns of a realistic dynamic graph, e.g., the node and edge distribution over time. Moreover, they do not generate realistic attributes. More robust statistical models, e.g., the aforementioned MAG model, can generate realistic dynamic graphs with attributes but still *no ground truth*.

To address the above shortcomings, we propose a methodology to generate realistic, dynamic-attributed, bipartite graphs with known instances of anomalies, as encompassed by **G2A2**, our *graph generator with attributes and anomalies*, which uses a hybrid "statistical + generative adversarial network" approach to achieve realism. Graph generation within G2A2 is asynchronous, making it more scalable and capable of generating millions of edges in under a minute. Table 1 summarizes how our approach compares and contrasts to the existing state of the art. To the best of our knowledge, our methodology is the first to generate a *synthetic graph that is realistic, dynamic-attributed, and bipartite with known instances of anomaly (i.e., ground truth)*. In all, our contributions are as follows:

- G2A2, a graph generator with attributes and anomalies that produces realistic, dynamic, attributed, bipartite graphs with known instances of anomalies.
  - The capturing of the cyclic pattern of nodes and edges over time as found in real-world applications.
  - A scalable approach to generate graphs in parallel.
  - A novel approach to inject anomalies with the properties of anomaly propagation and burstiness.
- A rigorous evaluation of the quality and computation time of our generated synthetic graph compared to three real-world graph datasets (i.e., Reddit [17], Wikipedia [17], and P-core network traffic [23]) with respect to three similarity criteria: graph similarity, anomaly similarity, and attribute similarity.
- A realistic graph library containing G2A2-generated social media graphs, article graphs, and Internet traffic graphs. The library encompasses graphs with varying anomaly percentages, as well as different numbers of nodes and edges, providing researchers with a tool for benchmarking various algorithms. The library can be accessed from here: https://github.com/vtsynergy/G2A2

The rest of the paper is organized as follows: §2 Observations, §3 Methodology, §4 Results and Evaluation, §5 Related Work, and §6 Conclusion and Future Work.

2 OBSERVATIONS

After conducting an in-depth analysis of real-world graph datasets, we identified four crucial observations that must be incorporated into synthetic graphs to ensure realism.

Observation 1. ***The node and edge probability distribution over time is cyclic.***

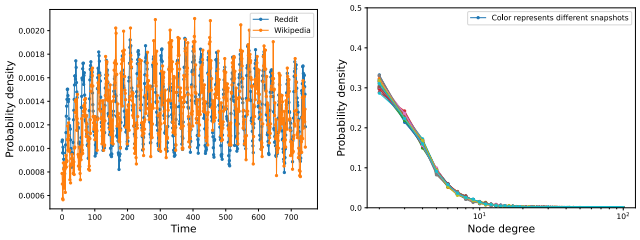
Previous attempts to generate dynamic graphs have focused on modeling the structural changes in graphs using properties such as densification power law and shrinking diameter [19]. However, we found that these generators fail to capture the seasonality or cyclic nature of node and edge time distribution found in many real-world dynamic application graphs (see Figure 2a). Several anomaly detection algorithms use the dramatic increase in node and edge count per unit of time to flag the associated node and edge as anomalous [3]; however, these algorithms are not tested on seasonality-based synthetic graphs, where they can result in many false positives.

Observation 2. ***The node degree distributions within snapshots are similar.***

For a given time t, we see a similar node degree distribution within all the graph snapshots of duration t, as shown in Figure 2b. We also observe that the higher-degree nodes have a relatively high occurrence throughout the time distribution regardless of the snapshot. That is, the relative ranking of node degrees stays similar throughout the time distribution.

Table 1: Comparison of graph-generation methods

Methods	Dynamic	Attributed	Graphs *NOT* required for training	Ground truth	Asynchronous
Deep Generative Graph Generation					
D2G2 [35]	✓	✓			
TagGen [36]	✓				
Statistical Graph Generation					
E-R [10]			✓		
Kronecker [18]	✓		✓		
MAG [15]	✓	✓	✓		
Our proposed methodology					
G2A2	✓	✓	✓	✓	✓



(a) Observation 1 **(b) Observation 2**

Figure 2: Visualization of observations 1 and 2.

Observation 3. ***Graph anomaly properties include burstiness and propagation.***

Several anomaly detection methods, e.g., [3, 30], annotate a certain percentage of data points as anomalies randomly or manually. However, graph anomalies exhibit distinct characteristics when compared to non-graph anomalies. They have structural and temporal properties such as burstiness and propagation, respectively, as shown in Figure 3.

Observation 4. ***"Edge + attributes" have better class separability compared to attributes alone.***

We visualize the class separability between anomalous and normal edges using t-SNE [27] and observe that the inclusion of edge embedding vastly enhances the separability. We use node2vec [12] for edge embedding and apply the Hammard operator. Once we obtain the edge embedding features, we concatenate them with the edge attributes. After dimension reduction using t-SNE, we observe their separability, as shown in Figure 4.

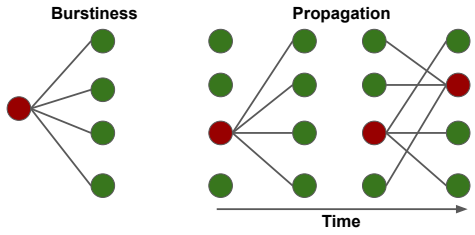
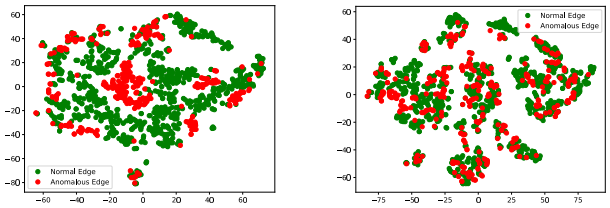


Figure 3: Example of graph burstiness and propagation.



(a) Edge Embedding + Attributes **(b) Attributes**

Figure 4: Visualization of edge embedding + attributes vs. just the attributes of Reddit dataset.

3 METHODOLOGY

Figure 5 illustrates our G2A2 methodology in three high-level steps: (1) dynamic bipartite graph generation, (2) anomaly injection, and (3) attribute generation and mapping. Our dynamic bipartite graph generation uniquely models time and degree distributions to generate realistic graphs. This is in contrast to existing graph generation methodologies that suffer from (a) modeling the entire graph under one distribution *without* considering the time distribution of the graph snapshots and (b) adding only noise (or outliers) to the graph rather than adding anomalies. Next, our novel anomaly injection algorithm emulates common graph-anomaly properties, such as burstiness [6] and propagation [32], in the generated graph. Then, by leveraging a generative adversarial network (GAN) model like CTGAN [29], we generate and map realistic attributes to the graph.

To explain our G2A2 methodology, we first define some notations for dynamic attributed bipartite graphs and anomalies as well as formulate our problem statement.

Definition 1. ***Dynamic Attributed Bipartite Graph.*** *An attributed bipartite graph is a graph $G(U, V, E, F)$, where U and V are two non-overlapping sets of nodes, E represents the edges connecting U and V (and there exists no edge within the sets themselves), and F denotes the edge attributes or features of the graph G. A dynamic or time-evolving attributed bipartite graph is a set of snapshots $G = \{G_1, G_2, G_3, ...G_t\}$ where $t \in T$, the total number of snapshots.*

Because a graph data structure possesses so many dependencies, *scaling* the graph generation process is challenging. In addition to capturing the realistic nature of graphs, as noted in Observations 1 and 2, modeling our dynamic graphs as a set of snapshots helps to *parallelize* and *scale* the generation process, as shown in Figure 6.

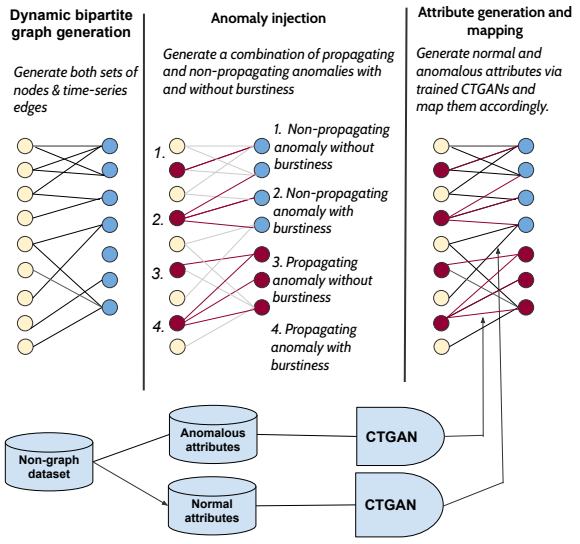


Figure 5: G2A2 methodology

Definition 2. ***Graph Anomalies.*** *Anomalies are rare occurrences or events in data that are often hard to detect. Given a graph G, anomalies constitute nodes or edges that are significantly different from the rest of the graph. In a graph G, we represent anomalies as $A = \{A_1, A_2, \ldots, A_r\}$ where r is the total number of injected anomalies. A^U and A^V are the sets of node anomalies, and A^e represents the edge anomalies.*

Unlike an outlier that is just noise, anomalies have patterns associated with real-world events [5]. For example, a distributed denial-of-service (DDoS) attack [13] is an anomaly. In the context of a graph, there exist three types of anomalies: (1) point-based, (2) time-based [5], and (3) graph-based [22]. Detecting point-based anomalies can occur independently without exploring their relationship with the other data points. Time-based anomalies exhibit patterns over time, which can be detected by plotting and viewing individual anomalies over time. Graph-based anomalies, however, require a deeper understanding of the entities involved and the relationships between them. They can only be detected when viewing the relationships within a graph.

Definition 3. ***Problem Statement.*** *Given a non-graph dataset D, how to generate a dynamic attributed bipartite graph $G(U, V, E, F)$ with T snapshots and a set of graph anomalies $A = \{A_1, A_2, ...A_r\}$? Graph G should have realistic statistical distributions (time and degree), and the attributes F should be similar to D.*

3.1 Dynamic Bipartite Graph Generation

To generate a realistic dynamic bipartite graph, we need to model both a realistic time distribution and degree distribution. As elaborated in Observation 1, most generative models do not consider the cyclic nature of the node and edge addition/removal found in many real-world dynamic application graphs. We capture this property using a probabilistic model to make our generation more realistic. Past studies show that Cauchy distributions provide a realistic cyclic representation of the time-series data [34]. While the degree

distribution is often assumed to follow the power law [9], recent studies show that the power-law distribution itself may be too narrow for modeling a realistic degree distribution. Instead, a more general distribution, such as gamma, is more suitable [2]. Therefore, we use the Cauchy distribution to model a realistic time distribution and the gamma distribution to model a realistic degree distribution. Below we explain how G2A2 generates a dynamic bipartite graph using the Cauchy and gamma distributions.

Time Distribution: In many real-world scenarios, the frequency of specific events changes over time and often follows a cyclic pattern. For example, the volume of Internet traffic increases during the day and falls overnight every 24 hours. Based on our experiments on diverse datasets across many domains, we found that the Cauchy distribution fits the best across such cyclic time patterns [1]. In addition, we found that the number of nodes participating in a graph snapshot varies cyclically with time, as shown in Observation 1. The Cauchy distribution is heavy-tailed [24], as shown in Equation (1):

$$f(x) = \frac{1}{(s\pi(1 + (x - l)s)^2)} \tag{1}$$

where l is the location parameter and s is the scale parameter.

The general idea is to use the Cauchy distribution to generate the time distribution for t hours and then repeat it T/t times, where T is the total number of graph snapshots. By combining all the T/t Cauchy distributions, we obtain the overall time distribution. We calculate three such time distributions for edges E, nodes U, and nodes V and then use these time distributions to sample nodes and calculate the number of edges for a single graph snapshot. Using the obtained edge count, we obtain the degree sequence of the sampled nodes via the gamma distribution.

To sample the nodes, we use weighted sampling [8], where the weight is inversely proportional to the node degree probability obtained from the gamma distribution. We then feed the degree sequence as input to the configuration model to get the final graph snapshot. This process iterates to generate all the graph snapshots in the graph, as articulated in Algorithm 1.

Algorithm 1: Dynamic Bipartite Graph Generation

Input: Cauchy (cPara) and gamma (gPara) parameters; total time (T); total number of U and V nodes ($|U|$, $|V|$); total number of edges ($|E|$)

Output: Dynamic Bipartite Graph (G)

$U^{count} = Cauchy(|U|, T, cPara)$;
$V^{count} = Cauchy(|V|, T, cPara)$;
$E^{count} = Cauchy(|E|, T, cPara)$;
$U^{probability} = Gamma(|U|, gPara)$;
$V^{probability} = Gamma(|V|, gPara)$;
for t *in* T **do**
  $U^{snapshot} = sampleNodes(U_t^{count}, U^{probability})$;
  $V^{snapshot} = sampleNodes(V_t^{count}, V^{probability})$;
  $seq^{snapshot} =$
   $sampleEdges(U^{snapshot}, V^{snapshot}, E_t^{count})$;
  $add(G_t, BiCM(seq^{snapshot}))$;
end

Degree Distribution: The degree distribution of a graph defines its primary structure. Similar to our experiments performed on real-world datasets to deduce time distributions, we found that most real-world graph degrees have a long-tailed distribution [18]. Out of the many different types of long-tailed distributions, the gamma distribution fits the best across real-world datasets. The gamma distribution is a two-parameter family of continuous probability distributions [24] represented by the following equation:

$$g(x) = \frac{(\frac{(x - l)}{s})^{a-1} \exp{(\frac{-(x - l)}{s})}}{s\Gamma(a)} \tag{2}$$

where a is the shape parameter, l is the location parameter, s is the scale parameter, and Γ is the gamma function, i.e.,

$$\Gamma(a) = \int_0^\infty t^{a-1} e^{-t} dt \tag{3}$$

We calculate the degree distribution of both U and V nodes separately as they are disjoint sets. Next, this probability degree distribution computes the degree sequences of both the U and V nodes of the graph snapshot. These sequences then serve as inputs to the bipartite configuration model (BiCM) [25], as shown in Algorithm 1.

Bipartite Configuration Model (BiCM): The BiCM generates a graph from a given degree sequence. The generated graph, in turn, possesses real-world graph properties, such as the small-world effect [28] and high clustering. We use the bipartite version of the configuration model (BiCM) [25], where we pass two degree sequences of U and V nodes, respectively, as inputs to the model. Each graph snapshot can be generated asynchronously, and we merge them together, in the end, such that the relative ranking of node degrees stays similar throughout the time, as elaborated in Observation 2. Figure 6 shows a visualization of our dynamic bipartite graph generation.

3.2 Anomaly Injection

Here we seek to inject a range of graph anomalies into our graph based on some of the most common properties of graph anomalies across domains, such as burstiness and propagation. In a graph, anomalies can occur at a node or an edge; while edge anomalies directly depend on the change in the state of the nodes.

For example, in network intrusion, when a system (represented by a node) is compromised, unknown or suspicious transactions are made from that system (represented by anomalous edges) that the system user would not make otherwise. Additionally, the change in the state of the node can happen due to either a graph-associated event that occurred at a previous time step or some other external factors. For instance, a system can get compromised by an attack that was propagated from a previous connection, or the system can be the original attacker itself [16]. The anomalous behavior differs between domains, too. In the case of social media fraud, anomalies do *not* propagate, and the frequency of their occurrence is less than that of a malicious system attacking other systems in a network graph to gain access. Given the range of potential anomalies that can occur in graphs, we define the following:

Definition 4. ***Attacking, Victim, and Infected Nodes.*** *Attacking nodes are anomalous nodes that initiate anomalous edges with the*

other nodes. Victim nodes are targeted by attacking nodes. If a victim node converts to an attacking node, it becomes an infected node.

Definition 5. ***Anomaly Subgraph.*** *An anomaly subgraph A_i is defined as a set of attacking nodes and victim nodes $A = \{A_1, A_2, \ldots, A_r\}$, where r is the total number of injected anomalies.*

Definition 6. ***Anomaly Burstiness.*** *If an anomalous node(s) attacks a victim node(s) with a high volume of edges, burstiness can be quantified as $\frac{|\{A^U, A^V\}|}{|\{U^{victim}, V^{victim}\}|}$, where $\{A^U, A^V\}$ represents the set of attacking nodes and $\{U^{victim}, V^{victim}\}$ represents the set of victim nodes.*

Definition 7. ***Anomaly Propagation.*** *Given an anomaly subgraph A, if there exists an edge between $U \in A$ and $V \notin A$ at time t_1, then there has been an event of anomaly propagation if at $t_2 > t_1$, $V \in A$.*

Algorithm 2 presents our anomaly injection algorithm, which accepts tunable parameters such as anomaly percentage, burstiness value, and anomaly duration. For anomalies without burstiness, we set the burstiness value to one. Additionally, we can enable propagation by setting a propagation flag and propagation ratio. Either or both U and V nodes can be anomalous based on the flag value. For propagation, both the U and V nodes must be anomalous. Figures 7 and 8 illustrate burstiness and propagation, respectively.

Algorithm 2: Anomaly Injection

Input: Dynamic bipartite graph ($G = \{U, V, E\}$), initial number of anomalous U and V nodes (c_{u_a}, c_{v_a}), anomaly percentage (ap), burstiness value (b), propagation ratio (p), anomaly duration (T_a)

Output: dynamic bipartite graph with anomalies

$A^U = sampleNodes(G, c_{u_a})$;
$A^V = sampleNodes(G, c_{v_a})$;
$U^{victim} = sampleNodes(G, |A^U| * b)$;
$V^{victim} = sampleNodes(G, |A^V| * b)$;
for *t in* T_a **do**
  $c_e = |E_t| * ap$;
  if *both U and V nodes can be anomalous* **then**
    $add(G_t, sampleEdges(A^U, V^{victim}, c_e/2))$;
    $add(G_t, sampleEdges(A^V, U^{victim}, c_e/2))$;
    if *Anomaly Propagation is true* **then**
      $add(A^U, sampleNodes(U^{victim}, |U^{victim}| * p))$;
      $add(A^V, sampleNodes(V^{victim}, |V^{victim}| * p))$;
    end
  end
  if *U can be anomalous* **then**
    $add(G_t, sampleEdges(A^U, V^{victim}, c_e))$;
  end
  if *V can be anomalous* **then**
    $add(G_t, sampleEdges(A^V, U^{victim}, c_e))$;
  end
end

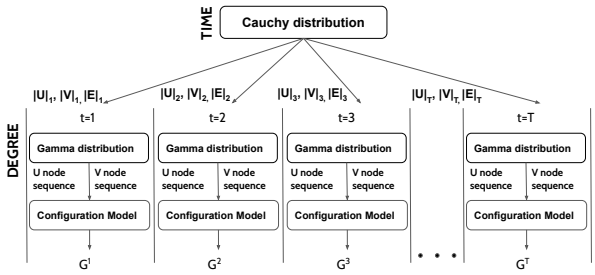


Figure 6: Dynamic bipartite graph generation

3.3 Attribute Generation and Mapping

The final step of our G2A2 methodology adds attributes to our generated dynamic bipartite graph, using the attributes from an existing non-graph dataset D. As mentioned in observation 4, edge embedding + attributes should be more separable than just the attributes. Therefore, we generate normal and anomalous attributes separately and map them onto respective edges.

We use two conditional tabular generative adversarial networks (CTGANs) [29] to generate synthetic attributes for our graph. Based on the ground truth, we divide the original dataset D into two parts: normal (D_n) and anomalous (D_a). In the absence of ground truth, we can apply clustering algorithms such as k-means with clusters equal to two. We train a CTGAN on each of D_n and D_a and map the generated attributes to normal and anomalous edges of our generated graph, respectively.

4 RESULTS AND EVALUATION

The goal of our work is to generate realistic, dynamic, attributed, bipartite graphs. However, evaluating these generated graphs remains a challenge [26]. There exists no single metric that can precisely quantify a generated graph's "realism" or its level of similarity to real-world graphs. Therefore, we propose a comprehensive three-step similarity evaluation approach that is analogous to our three-step generation methodology in order to assess our generated graph's level of similarity at the graph, anomaly, and attribute levels compared to real-world graph datasets. In our three-step evaluation approach, we compare our generated graph against three real-world graph datasets, namely P-core [23], Reddit [17], and Wikipedia [17], based on three similarity criteria: *(1) graph similarity*, *(2) anomaly similarity*, and *(3) attributes similarity*.

4.1 Experimental Setup

To quantify the similarities, we leverage past research, including GraphRNN [31] and GRAN [21], which evaluate the performance of a model's graph generation by comparing the similarity between various graph statistics of the generated and ground truth graphs, e.g., degree distribution and clustering coefficient distribution using the maximum mean discrepancy (MMD) metric. The following subsections explain the three similarity criteria and the graph statistics indicative of each similarity criteria.

4.1.1 ***Graph similarity:*** In our graph similarity evaluation, we quantitatively compare graph statistics such as (1) the time distribution, i.e., nodes and edges distributed across time, (2) the degree

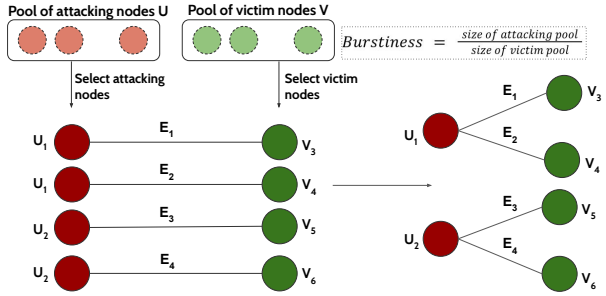


Figure 7: Anomaly burstiness

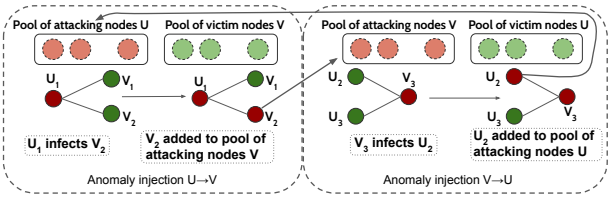


Figure 8: Anomaly propagation

distribution, and (3) the bipartite clustering coefficient (BCC) distribution [33] of our generated graph against the three real-world graphs using the MMD metric. The degree distribution of a graph is a probability distribution, given by Equation (4):

$$P_{degree}(k) = \frac{|\{i|degree(i) = k\}|}{N} \tag{4}$$

where N is the total number of nodes.

The time distribution of a graph is a probability distribution of the number of nodes and edges over time. We compute the node time distribution of U and V separately. The node time distributions of U and V and the edge time distribution of a graph can be represented via Equation (5):

$$P^{edge}_{time}(t) = \frac{|E_t|}{T} \qquad P^{node_U}_{time}(t) = \frac{|U_t|}{T} \qquad P^{node_V}_{time}(t) = \frac{|V_t|}{T} \tag{5}$$

where T is the total number of snapshots.

The bipartite clustering coefficient (BCC) of a graph is a measure of the local density of interactions. The BCC of a node can be calculated using the following formula:

$$c_u = \frac{\sum_{v \in N(N(u))} c_{uv}}{|N(N(u))|} \tag{6}$$

where $N(N(u))$ are the second-order neighbors of u in G excluding u, and c_{uv} is the pairwise clustering coefficient between nodes u and v. For our experiments, we define c_{uv} using the following equation:

$$c_{uv} = \frac{|N(u) \cap N(v)|}{|N(u) \cup N(v)|} \tag{7}$$

We calculate the average bipartite clustering coefficient as:

$$BCC_X = \frac{1}{|X|} \sum_{v \in X} c_v \tag{8}$$

where $X \in \{U, V\}$.

4.1.2 ***Anomaly similarity:*** We quantify anomaly similarity separately from graph similarity because anomalies are rare occurrences and typically do not affect the overall graph statistics. To compare the anomalies, we filter the anomaly subgraphs from both the generated graph and the real-world graphs. Similar to graph similarity, we then compare the BCC distribution of the two anomaly subgraphs using the MMD metric.

4.1.3 ***Attributes similarity:*** To quantify the similarity of attributes between our generated and real-world graphs, we check if both sets of attributes have been drawn from the same distribution and use the MMD metric to verify the distribution similarity [11]. To do so, we preprocess the data by normalizing it and then create a histogram with a specific bin size, which we fixed to 100 here.

4.2 Real-world Graph Datasets

For our evaluation, we use three real-world datasets: P-core, Reddit, and Wikipedia. P-core is a dataset provided by Advanced Research Computing (ARC) at Virginia Tech (VT) [23]. Reddit and Wikipedia are from the JODIE repository [17]. Below is a qualitative description of the datasets, followed by a quantitative summary in Table 2.

- **P-core:** This dataset contains network traffic flows (E) information going through the edge server from the VT domain (U) to the rest-of-the-world (RoW) domain (V) and vice versa. The anomaly (A^e) for this dataset is a staged Mirai-botnet attack.
- **Reddit:** This dataset contains posts (E) made by users (U) on subreddits (V) in a month (T). Anomaly labels (A^e) at the edge level represent the interactions after which the user got banned.
- **Wikipedia:** This dataset contains edits (E) done by users (U) on Wikipedia pages (V) recorded over a month. Additionally, anomaly labels (A^e) at the edge level represent the last edit made by the user before they got banned.

Table 2: Summary of real-world graph datasets

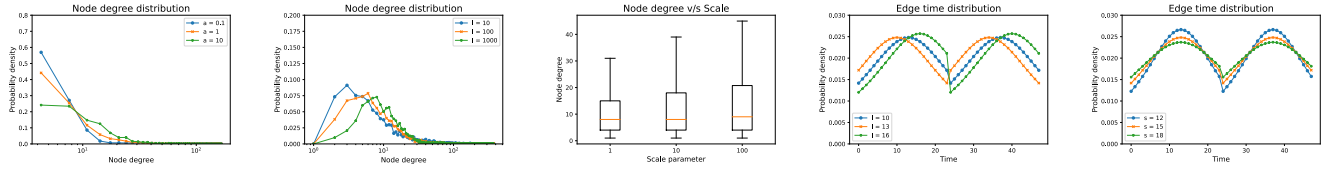
Dataset	$\|U\|$	$\|V\|$	$\|E\|$	$\|A^e\|$	$\|F\|$	T(hours)
P-core	157225	96037	597098	6012 (1%)	14	48
Reddit	10000	984	672447	366 (0.05%)	172	744
Wikipedia	8227	1000	157474	217 (0.14%)	172	744

4.3 Generating Realistic Graphs

By setting the parameters in G2A2 appropriately, we can generate realistic graphs that are similar to real-world graphs with respect to graph similarity, anomaly similarity, and attribute similarity. In addition, we compare G2A2 to other state-of-the-art methodologies to show that G2A2 outperforms them by reducing the MMD metric over various graph distributions and computation time.

4.3.1 ***Setting Cauchy and gamma parameters:*** While determining the exact values of the Cauchy and gamma distribution parameters to generate a realistic graph may be difficult, it is easier to estimate them based on their significance in shaping the overall desired distribution (a distribution similar to that of a real-world graph). For instance, the Cauchy distribution parameter location (l) and scale (s) determine the position of the peak and length of peak to trough, respectively.

Similarly, for the gamma distribution, we have three parameters: shape (a), location (l), and scale (s). The shape (a) determines the



(a) Gamma: a **(b) Gamma: l** **(c) Gamma: s** **(d) Cauchy: l** **(e) Cauchy: s**

Figure 9: Parameter analysis of gamma (a, b, c) on the node degree, and Cauchy (d, e) on edge time distribution.

Table 3: Comparison of G2A2 with the other graph generation models. For all MMD results, lower is better.

Methods	P-core					Reddit					Wikipedia				
	Graph Similarity			Anomaly	Attribute	Graph Similarity			Anomaly	Attribute	Graph Similarity			Anomaly	Attribute
	Degree	BCC	Time	Similarity	Similarity	Degree	BCC	Time	Similarity	Similarity	Degree	BCC	Time	Similarity	Similarity
E-R	1.047	0.957	NA	NA	NA	0.750	0.966	NA	NA	NA	0.670	0.911	NA	NA	NA
Kronecker	0.578	0.352	0.789	NA	NA	0.539	0.347	0.616	NA	NA	0.511	0.398	0.675	NA	NA
Kronecker + attributes	0.578	0.352	0.789	NA	0.584	0.539	0.347	0.616	NA	0.712	0.511	0.398	0.675	NA	0.649
MAG	1.778	0.544	NA	NA	1.047	1.686	0.541	NA	NA	1.854	1.725	0.522	NA	NA	1.959
G2A2	**0.128**	**0.194**	**0.003**	**0.064**	**0.076**	**0.142**	**0.252**	**0.016**	**0.165**	**0.044**	**0.150**	**0.210**	**0.020**	**0.255**	**0.010**

skewness of the degree distribution. The lower the shape parameter, the more skewed the graph is. The location parameter (l) specifies the minimum probability of the smallest node degree. The higher the location parameter, the greater the minimum node degree. The gamma distribution's scale parameter (s) determines the difference between the lowest degree and highest degree. The higher the scale parameter, the more significant the difference. Figure 9 shows the effect of setting the gamma and Cauchy function parameters.

4.3.2 ***G2A2-generated graphs vs. real-world graphs:*** G2A2 generates graphs similar to the following three real-world graphs: P-core, Reddit, and Wikipedia. We evaluate their quality by plotting and comparing the degree and time distribution (node + edge) of the generated graph versus the real graph, as shown in Figure 10. Parameters used for the generation have been obtained heuristically, as explained in the previous section, and do not represent the optimal value. The objective is to demonstrate the capability of G2A2 in generating realistic graphs even without the optimal values of parameters in place.

An important note is that when we generate graphs for the social media graphs (Reddit) and article graphs (Wikipedia), the propagation flag for the anomaly injection is set to false. This means that the victim nodes do not participate in the attack. In contrast, the propagation flag is set to true when generating graphs for the Internet traffic graph (P-core). This means the victim nodes participate in the attack if they are infected.

4.3.3 ***G2A2 vs. the other models:*** We compare the quality of the generated graphs via G2A2 against the ones generated by existing models, such as Erdős-Rényi (E-R) [10], Kronecker [18], and Multiplicative Attribute Graph (MAG) [15]. Note that we have not compared our model with any of the deep generative models such as D2G2 [35] and TagGen [36] because of insufficient training data which is also one of the key problems that we have highlighted with regard to deep generative methods earlier as shown in Figure 1. We have also included Kronecker with randomly generated attributes as a baseline to show how our model compares to randomly generated attributes. However, all of these existing models could only generate parts of a dynamic attributed bipartite graph with instances of anomalies. Therefore, we could only compare the relative parts of the graph these models can generate with G2A2-generated graphs.

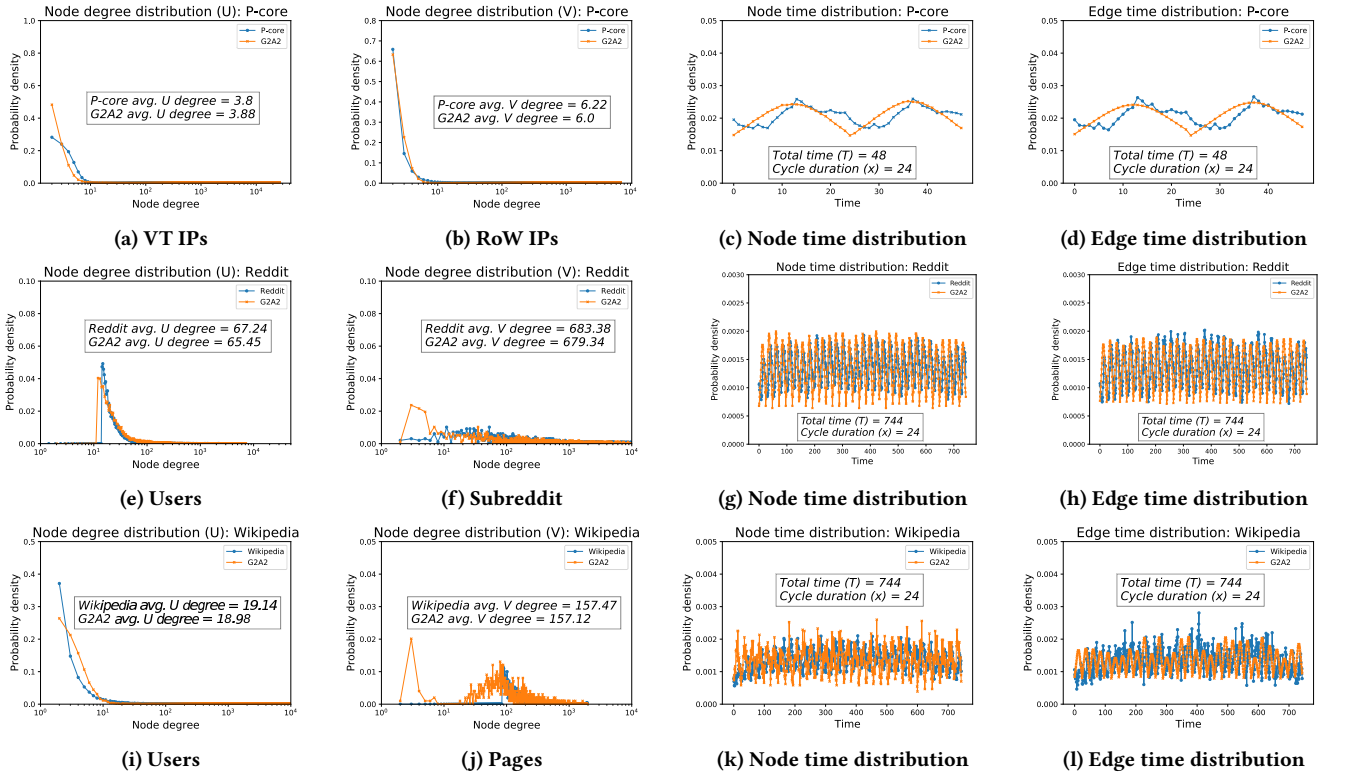
The Kronecker graph generator performs the best of all the existing models that we compared G2A2 with. The Kronecker graph generator uses an initiator matrix with a given pattern and can generate bipartite graphs. We obtain the initiator matrix using Kroneckerfit (provided by the SNAP library [20]) that can learn parameters for the model. Kronecker can also generate realistic static and dynamic properties of a graph. However, the generator does *not* provide a timestamp to the edges; we had to equally divide the edges into different timestamps to evaluate the temporal component of the graph.

We also evaluated G2A2 with a modified version of Kronecker by generating a random matrix with the attribute dimension's size and mapping it to the edges of the Kronecker graph. As shown in Table 3, G2A2 delivers the best results, followed by Kronecker+attributes.

4.3.4 ***Computation time:*** Figure 11 shows the computation time of G2A2 vs. other baseline models across a varying number of edges, number of nodes, and number of snapshots generated. We only measure the computation time of dynamic bipartite graph generation to keep the comparison fair. The experiments ran on a 2.10-GHz Intel(R) Xeon(R) Gold 6130 CPU.

G2A2's single-core performance is on par with Kronecker, even though G2A2 generates graphs of higher complexity. We found that the G2A2 computation time when using only one core is similar to the fastest graph generator algorithm.

A notable advantage of G2A2 is that the graph snapshots can be computed independently, allowing us to naturally parallelize the graph generation. Figure 11d shows that G2A2 delivers nearly perfect linear speed-up. The maximum speed-up depends on the total number of snapshots. That is, if we have eight million edges to generate but only a single snapshot, the computation proceeds serially. In contrast, if we have sixteen million edges and one hundred snapshots, we can achieve up to a 100-fold speed-up.

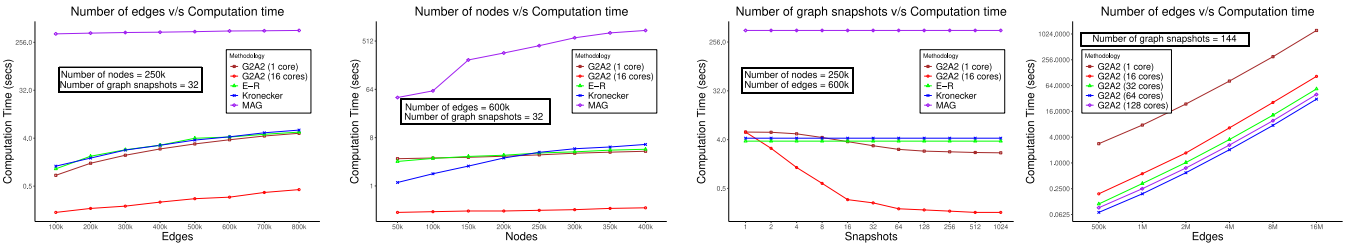


(a) VT IPs (b) RoW IPs (c) Node time distribution (d) Edge time distribution

(e) Users (f) Subreddit (g) Node time distribution (h) Edge time distribution

(i) Users (j) Pages (k) Node time distribution (l) Edge time distribution

Figure 10: G2A2-generated vs. real-world graphs: P-core (a, b, c, d), Reddit (e, f, g, h) and Wikipedia (i, j, k, l)



(a) Edge analysis (b) Node analysis (c) Snapshot analysis (d) Large graph generation

Figure 11: Computation time of G2A2 vs. other models (a,b,c) and computation time of G2A2 for large graph generation (d)

5 RELATED WORK

In addition to the work mentioned previously in §1, we discuss a few other approaches relevant to graph generation and categorize graph generation methodologies into two classes: statistical graph generation and deep generative (DG) graph generation.

5.1 Statistical Graph Generation

Many of the traditional methods of synthetic graph generation involve random graph modeling. The most popular model is the Erdős-Rényi (E-R) model [10]. Unfortunately, the resultant graphs have minimal utility in terms of real-world applications because the E-R model assumes the same probability for every edge in the graph and generates an approximate Poisson distribution. Studies have found that most of the real-world graphs have a small-world effect [28] and exhibit a power-law distribution [4], i.e., a skewed distribution. The bulk of such studies focus on static graphs but with increasing interest in time-evolving graphs. Work like the forest fire [19] presents the characteristics of a time-evolving graph, such as the densification power laws and shrinking diameters. However, there has been a limited effort in highlighting the distribution of the participating nodes and edges over time.

5.2 Deep Generative (DG) Graph Generation

Deep generative (DG) graph generation has gained significant traction [31, 35]. The advantages of DG methodologies over statistical ones are at least two-fold. First, DG methodologies learn the intrinsic properties of the graph without explicitly mentioning them.

Second, these models can inherently learn attributes (node or edge). On the other hand, a disadvantage with these models is that they require a *lot* of data, for which there is a dearth of such publicly available datasets. Moreover, a majority of DG models focus only on static graphs [14] and fail to generate ground truth from the training graphs.

6 CONCLUSION AND FUTURE WORK

This paper introduces a new methodology to generate realistic, dynamic-attributed bipartite graphs with anomalies. Our G2A2 methodology can generate graphs with a realistic degree and time distribution, multiple types of graph anomalies, and attributes similar to an existing non-graph dataset. In addition, G2A2 can easily be parallelized for faster computation and can generate realistic graphs across multiple domains. We also release a graph library that contains generated graph datasets with ground truth from various domains that can be used by researchers to benchmark their algorithms. The library can be accessed from here: https://github.com/vtsynergy/G2A2.

In the future, we plan to extend G2A2 to generate graphs with node attributes as well. This will enable researchers to test their anomaly detection algorithms for an even wider range of problems.

7 ACKNOWLEDGEMENT

The work was supported in part by NSF I/UCRC CNS-1822080 via the NSF Center for Space, High-performance, and Resilient Computing (SHREC). The authors also acknowledge Advanced Research Computing at Virginia Tech for providing computational resources and technical support that have contributed to the results reported within this paper.

REFERENCES

[1] A. Alzaatreh, Carl Lee, F. Famoye, and Indranil K. Ghosh. 2016. The generalized Cauchy family of distributions with applications. *Journal of Statistical Distributions and Applications* 3 (2016), 1–16.

[2] L. Benguigui and M. Marinov. 2015. A classification of natural and social distributions Part one: the descriptions. arXiv:1507.03408 [physics.soc-ph]

[3] S. Bhatia, R. Liu, B. Hooi, M. Yoon, K. Shin, and C. Faloutsos. 2022. Real-Time Anomaly Detection in Edge Streams. *ACM Trans. Knowl. Discov. Data* 16, 4 (2022), 22 pages. https://doi.org/10.1145/3494564

[4] M. Chakrabarti, L. Heath, and N. Ramakrishnan. 2017. New methods to generate massive synthetic networks. arXiv:1705.08473 [cs.SI]

[5] V. Chandola, A. Banerjee, and V. Kumar. 2009. Anomaly detection: A survey. *ACM Comput. Surv.* 41 (2009), 15:1–15:58.

[6] Z. Chen and A. Sun. 2020. Anomaly Detection on Dynamic Bipartite Graph with Burstiness. In *2020 IEEE International Conference on Data Mining (ICDM)*. IEEE, Sorrento, Italy, 966–971. https://doi.org/10.1109/ICDM50108.2020.00110

[7] K. Ding, J. Li, R. Bhanushali, and H. Liu. 2019. *Deep Anomaly Detection on Attributed Networks*. Proceedings of the 2019 SIAM International Conference on Data Mining, Canada, 594–602. https://doi.org/10.1137/1.9781611975673.67

[8] P. S. Efraimidis and P. G. Spirakis. 2006. Weighted random sampling with a reservoir. *Inf. Process. Lett.* 97 (2006), 181–185.

[9] N. Eikmeier and D. F. Gleich. 2017. Revisiting Power-law Distributions in Spectra of Real World Networks. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, New York, NY, USA, 817–826. https://doi.org/10.1145/3097983.3098128

[10] P. Erdös and A. Rényi. 1959. On Random Graphs I. *Publicationes Mathematicae Debrecen* 6 (1959), 290.

[11] A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schölkopf, and A. Smola. 2012. A Kernel Two-Sample Test. *J. Mach. Learn. Res.* 13 (2012), 723–773.

[12] A. Grover and J. Leskovec. 2016. node2vec: Scalable Feature Learning for Networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (San Francisco, California, USA). ACM, New York, NY, USA, 855–864. https://doi.org/10.1145/2939672.2939754

[13] Q. Gu and P. Liu. 2012. *Denial of Service Attacks*. Vol. 3. John Wiley and Sons, online, 454–468. https://doi.org/10.1002/9781118256107.ch29

[14] X. Guo and L. Zhao. 2020. A Systematic Survey on Deep Generative Models for Graph Generation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 45 (2020), 5370–5390.

[15] M. Kim and J. Leskovec. 2010. Multiplicative Attribute Graph Model of Real-World Networks. In *Algorithms and Models for the Web-Graph*, R. Kumar and D. Sivakumar (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 62–73.

[16] C. Kolias, G. Kambourakis, A. Stavrou, and J. Voas. 2017. DDoS in the IoT: Mirai and Other Botnets. *Computer* 50 (2017), 80–84.

[17] S. Kumar, X. Zhang, and J. Leskovec. 2019. Predicting Dynamic Embedding Trajectory in Temporal Interaction Networks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, New York, NY, USA, 1269–1278. https://doi.org/10.1145/3292500.3330895

[18] J. Leskovec, D. Chakrabarti, J. Kleinberg, and C. Faloutsos. 2005. Realistic, Mathematically Tractable Graph Generation and Evolution, Using Kronecker Multiplication. In *Knowledge Discovery in Databases*. Springer, Berlin, 133–145.

[19] J. Leskovec, J. Kleinberg, and C. Faloutsos. 2005. Graphs over time: densification laws, shrinking diameters and possible explanations. In *Proceedings of the 11th ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*. ACM, New York, NY, USA, 177–187. https://doi.org/10.1145/1081870.1081893

[20] J. Leskovec and R. Sosič. 2016. SNAP: A General-Purpose Network Analysis and Graph-Mining Library. *ACM Transactions on Intelligent Systems and Technology (TIST)* 8, 1 (2016), 1.

[21] R. Liao, Y. Li, Y. Song, S. Wang, W. L. Hamilton, D. Duvenaud, R. Urtasun, and R. Zemel. 2019. Efficient graph generation with graph recurrent attention networks. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, Vol. 383. Curran Associates Inc., Red Hook, NY, USA, 11 pages.

[22] X. Ma, J. Wu, S. Xue, J. Yang, C. Zhou, Q. Z. Sheng, H. Xiong, and L. Akoglu. 2023. A Comprehensive Survey on Graph Anomaly Detection With Deep Learning. *IEEE Transactions on Knowledge and Data Engineering* 35, 12 (Dec 2023), 12012–12038. https://doi.org/10.1109/TKDE.2021.3118815

[23] A. Nottingham, M. Gardner, J. Collyer, J. Lang, M. Veeraraghavan, M. Buchanan, J. Hiser, and J. Davidson. 2019. P-CORE Dataset.

[24] National Institute of Standards and Technology. 2001. *Security Requirements for Cryptographic Modules*. Technical Report Federal Information Processing Standards Publications 140-2, Change Notice 2 Dec. 03, 2002. U.S. Department of Commerce, Washington, D.C. https://doi.org/10.6028/nist.fips.140-2

[25] F. Saracco, R. Di Clemente, A. Gabrielli, and T. Squartini. 2015. Randomizing bipartite networks: the case of the World Trade Web. *Scientific Reports* 5 (2015), 18 pages.

[26] L. Theis, A. v. d. Oord, and M. Bethge. 2016. A note on the evaluation of generative models. In *4th International Conference on Learning Representations*. ICLR, Caribe Hilton, USA, 10 pages.

[27] L. van der Maaten and G. E. Hinton. 2008. Visualizing Data using t-SNE. *Journal of Machine Learning Research* 9 (2008), 2579–2605.

[28] D. J. Watts and S. H. Strogatz. 1998. Collective dynamics of 'small-world' networks. *Nature* 393, 6684 (1998), 440–442. https://doi.org/10.1038/30918

[29] L. Xu, M. Skoularidou, A. Cuesta-Infante, and K. Veeramachaneni. 2019. Modeling Tabular data using Conditional GAN. In *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (Eds.), Vol. 32. Curran Associates, Inc., Vancouver, Canada.

[30] M. Yoon, B. Hooi, K. Shin, and C. Faloutsos. 2019. Fast and Accurate Anomaly Detection in Dynamic Graphs with a Two-Pronged Approach. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, New York, USA, 647–657. https://doi.org/10.1145/3292500.3330946

[31] J. You, R. Ying, X. Ren, W. L. Hamilton, and J. Leskovec. 2018. GraphRNN: Generating Realistic Graphs with Deep Auto-regressive Models.. In *ICML (Proceedings of Machine Learning Research, Vol. 80)*. PMLR, Stockholm, Sweden, 5694–5703.

[32] W. Yu, W. Cheng, C. C. Aggarwal, K. Zhang, H. Chen, and W. Wang. 2018. NetWalk: A Flexible Deep Embedding Approach for Anomaly Detection in Dynamic Networks. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. Association for Computing Machinery, New York, NY, USA, 2672–2681. https://doi.org/10.1145/3219819.3220024

[33] P. Zhang, J. Wang, X. Li, M. Li, Z. Di, and Y. Fan. 2008. Clustering coefficient and community structure of bipartite networks. *Physica A: Statistical Mechanics and its Applications* 387, 27 (2008), 6 pages. https://doi.org/10.1016/j.physa.2008.09.006

[34] S. Zhang, M. Tao, X. Niu, and F. Huffer. 2020. Time-Varying Gaussian-Cauchy Mixture Models for Financial Risk Management. arXiv:2002.06102 [stat.AP]

[35] W. Zhang, L. Zhang, D. Pfoser, and L. Zhao. 2021. *Disentangled Dynamic Graph Deep Generation*. Proceedings of the 2021 SIAM International Conference on Data Mining (SDM), virtual, 738–746. https://doi.org/10.1137/1.9781611976700.83

[36] D. Zhou, L. Zheng, J. Han, and J. He. 2020. A Data-Driven Graph Generative Model for Temporal Interaction Networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, New York, USA, 401–411. https://doi.org/10.1145/3394486.3403082

[37] Y. Zhu, Y. Du, Y. Wang, Y. Xu, J. Zhang, Q. Liu, and S. Wu. 2022. A Survey on Deep Graph Generation: Methods and Applications. In *Proceedings of the First Learning on Graphs Conference (Proceedings of Machine Learning Research, Vol. 198)*. PMLR, virtual, 47:1–47:21.