Anomaly Detection in Graph: Unsupervised Learning, Graph-based Features and Deep Architecture

Dmitry Vengertsev, Hemal Thakkar, *Department of Computer Science, Stanford University*

***Abstract*—The ability to detect anomalies in a network is an increasingly important task in many applications. In this paper a combination of graph features and unsupervised learning methods is used to tackle anomaly detection problem in a multi-attributed graph. The focus is on special types of anomalies: global, neighbor-based, and community-based. We propose an algorithm for generating a graph that contains non-overlaping anomaly types. Synthetically generated anomalous graphs are analyzed with two graph-based anomaly detection methods: Direct Neighbour Outlier Detection Algorithm (DNODA); Community Neighbour Algorithm (CNA), and two unsupervised learning techniques: Isolation Forest and Deep Autoencoders.**

Keywords: Anomaly Detection, Graph Anomaly Synthesis, Isolated Forest, Deep Autoencoders

I. Introduction

Anomaly Detection refers to the problem of identifying patterns in data which do not conform to an expected behavior. Anomaly detection is applied to several domains like credit card fraud (Anomalous transactions), Network Security Breach (Anomalous Packets Data), Industrial Safety and Monitoring (Anomalous Sensor readings) and many more to retrieve vital, actionable data.

To tackle the anomaly detection problem, many techniques have been developed in the past decades [1], especially for spotting outliers and anomalies in unstructured collections of multi-dimensional data points. For the anomalies in the graph data structures however, data objects cannot be treated as points lying in a multi-dimensional space independently. Indeed, graph data exhibit inter-dependencies which should be accounted for during the anomaly detection process. There are several extensive studies to detect anomalies for both static and dynamic network topologies [2], [3].

II. Prior Work and Literature Review

OddBall algorithm [8] is widely used to detect anomaly in a network. The idea is to construct an induced sub-graph of neighboring nodes around each node(the ego), and then for the egonet estimate several features. Unfortunately, for our application OddBall is not applicable, since we have a network with fixed edges and time-varying attributes.

In [6], a network with time-varying edges is used to detect an unexpected road event. Two types of anomalies are discussed: anomalous weight of a graph's edge, and a significant anomalous region. Spatial-temporal correlation between edge weights were used for anomaly detection. Node attributes assumed to have constant values, and is not applicable for our problem.

Another paper that studies anomaly detection using locality statistics is [7], where the problem again is to detect anomaly in time series of graphs with time-dependent edges and fixed nodes' attributes.

One of the papers that deal with time-varying node attributes [4] introduces a method to detect anomalies in measurements collected by sensor systems. However, only global type of anomalies are considered, and network features were not used.

Another paper that deals with time-varying node attributes is [5], where the spatial and temporal correlations in the data are used to distinguish sensor failures from valid observations. The main application is to identify failure of cheap sensors in harsh environment, not to detect anomalous phenomenon like change of air molecules concentration that can cause a defect in a semiconductor fab.

In this paper we consider three types of anomalies: global (type one), neighbor-based (type two) and community-based anomalies (type three). Since there are no labeled dataset for graph anomaly detection, in this paper we describe methods to generate different types of anomalies in a graph. Then, using synthetic dataset, we compare different algorithms - graph-based, unsupervised learning and their combinations. And then, we used the best performing methods to label real world Intel lab dataset.

A. *Anomaly Detection using Graph Features*

For the analysis of type two anomalies, Direct Neighbour Outlier Detection Algorithm (DNODA) [9] approach is used. Intuitively, in this method, node is considered to be anomalous if it's attributes are significantly different from neighbors' attributes. In [9], unweighted static network is considered, we adapt their neighbor based anomaly score using time information and edge weights to detect an anomaly.

For the detection of type three anomalies (community outliers), two-stage algorithm is used: first, partition the network into several communities using network information [10], and then within each community, identify outliers based on the object information and time. This algorithm is referred to as Community Neighbor Algorithm (CNA).

The pattern-based anomalies (type two and type three) are suitable for interpretation and amenable for post-analysis by domain experts to reveal the root cause of anomaly.

davenger@stanford.edu, hemal@stanford.edu

B. Anomaly Detection using Unsupervised Learning Approaches

At this subsection methods for detecting global outliers, are considered. In [9], for global outliers detection they used density based method - Local Outlier Factor (LOF) [18]. However, we plan to use Isolation Forest [12] - state-of-the-art anomaly detection method - which is from the class of new methods that isolate anomalies, rather then building model for normal values (for details refer to next section). At first stage we will apply Isolation Forest to node attribute values, just to construct Ground Truth labels. But our main contribution here, is to apply this method for augmented node attribute data by graph-features with information of averaged neighboring attributes (DNODA). With taking into account connectivity information, we hope to identify type 2 anomalies as well. Next, we apply Deep Learning architecture, in particular, Deep Autoencoders [14]. Autoencoders are used to build a model for unsupervised data, and anomaly is defined as a point with the highest reconstruction error.

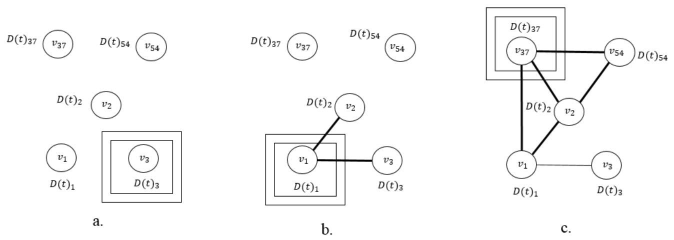


Fig. 1: Types of anomalies: a. Global anomaly - attributes of node v_3 are anomalous compare to other attributes in the whole network; b. Neighbor Anomaly - attributes of node v_2 are anomalous compare to it's neighbors v_2 and v_3; c. Community Anomaly - attributes of node v_{37} are different from same community nodes v_{54}, v_2, v_1

III. Methodology

A. Synthetic Dataset

It is a difficult task to obtain labelled anomaly data for Wireless Sensor networks, however we needed a baseline dataset to evaluate our algorithms and fine tune the parameters. We used the following techniques to generate synthetic data for different anomaly types.

Type 1 anomaly (Global) Generate n i.i.d random vectors such that each coordinate distributed according to a Pareto distribution with random sign. Due to the skew nature of Pareto distribution, it produces anomalies. We label a point as anomaly, when the joint probability of the point vector of a node is of the order of δ/n, for some constant $\delta > 0$.

Type 2 anomaly (Neighbour) and Type 3 anomaly (Community)

For Neighbour anomalies we require attributed graph data that is globally undeviating but has local inconsistencies that can be detected. To simulate this we applied the following technique:

Algorithm 1 Anomalous Graph Construction

```
1)  Generate $n$ i.i.d data points in $R^d$ uniformly distributed in $U(0, N)$
2)  For each $i \in n$
      • Let $D(i) \in R^d$ be the feature vector
      • Identify $k$ nearest neighbour in feature space
3)  Connect the nearest neighbours such that a fully connected graph is created
4)  Once the graph is built, identify open areas in the feature space, these areas consists of data points which fall under the same distribution but in the given graph do not have any data points or nearby neighbours. Let this set of data points be $P$.
5)  $P$ represents co-ordinates where local perturbations/noise can be injected artificially.
6)  $\forall p \in P$ identify Nearest neighbour $\hat{i} \in n$ such that $d(p, \hat{i})$ is maximum.
7)  Find $\hat{p}$ such that if $D(\hat{i}) = D(\hat{p})$ the underlying graph structure of $k$ nearest neighbours does not change.
8)  Thus $\hat{p}$ is not a global anomaly but is locally inconsistent
9)  Repeat steps $7, 8$ to identify to identify all $\hat{p} \in \hat{P}$
10) Repeat steps 1 to 9 for a minimum of $n^2$ times to obtain a large number of random graphs within the same distribution and $kNN$ relationship
11) Sort the list in descending order of total perturbation $\sum d(\hat{p}, \hat{i})$ and pick the top $x$ number of datasets for evaluation.
```

Fig 2 represents a graph with $n = 100, d = 2, k = 4$ and $N = 100$. The node position on the figure represents position in feature space. Green nodes show the random graph created by Steps 1 to 4. Red nodes represent the nodes perturbed as per steps 5 to 9. The Red nodes are labeled as anomalies.

For Community anomaly, similar technique is applied to identify random graphs wherein the graph consists of n i.i.d data points in R^d uniformly distributed in $U(0, N)$. A graph which has varying size of community diameters with outliers existing towards the boundaries and community intersections.

Fig 3 represents an example where there are no local inconsistencies, however there are outliers within a community. Examples for anomalous Nodes are: $23, 46$

B. Intel Lab Dataset

Origin and Data Statistics:

As a benchmark dataset for testing and method comparison we have selected Intel Lab data set [13]. This data set is represented as time-varying weighted multi-attributed graph that corresponds to sensors work for period from February 28th and April 5th, 2004 with properties shown in below table:

Property	Value
Number of nodes $\|V\|$	54
Max. Number of Edges $\|E\|$	2971
Feature Vector $D(t)$	$d = 3$
Total readings	2.3 million

Fig. 2: Synthetic Local Anomalies(Spikes)

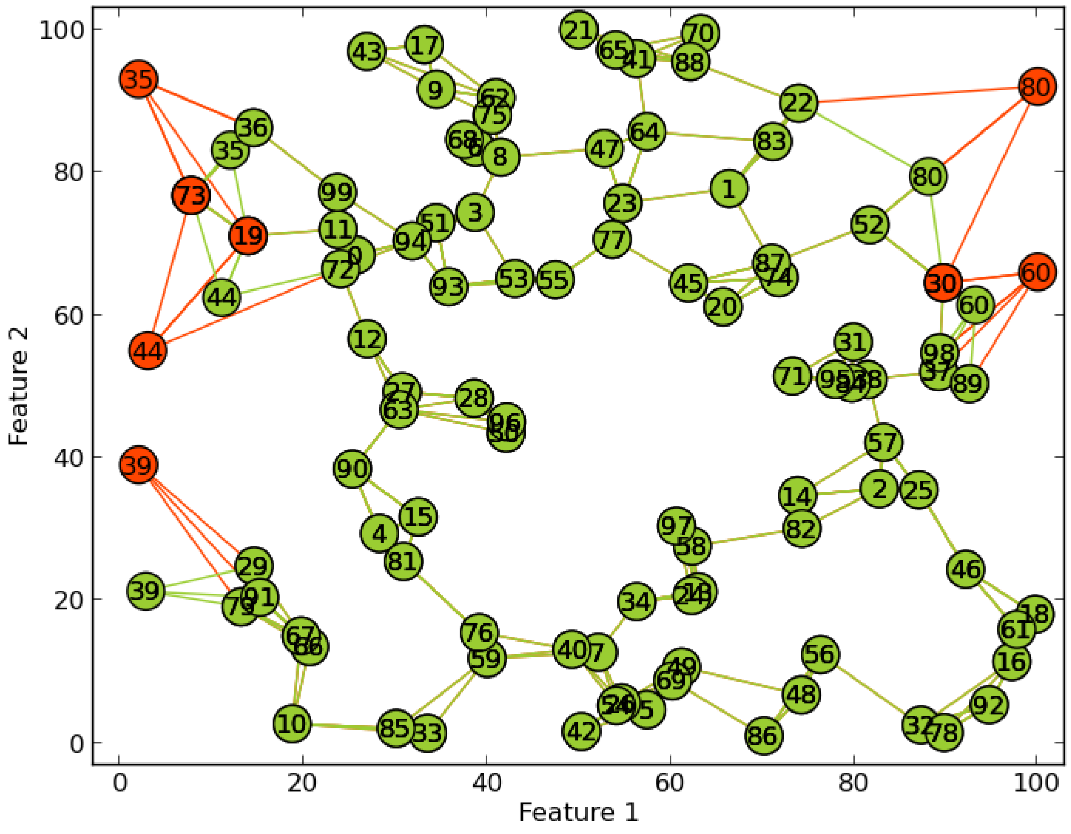
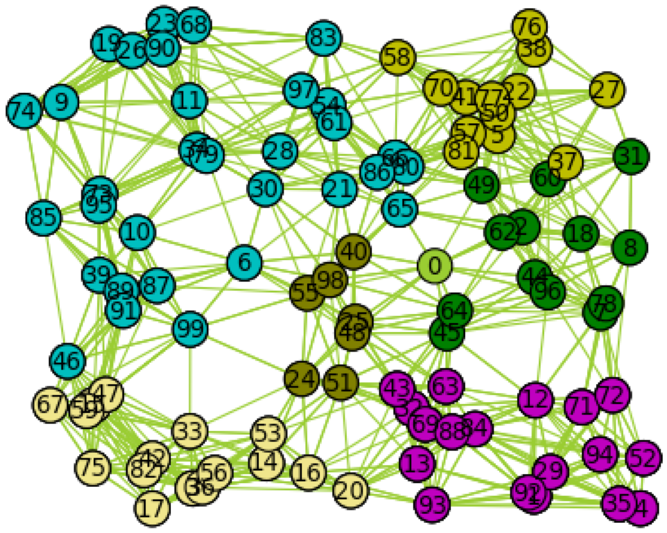


Fig. 3: Synthetic Communities Anomalies(Spikes)



[Note that the Nodes (or Mote represented by MoteIds) are Wireless Sensors continuously sensing and transmitting vital data]

Noise Reduction: The Wireless Sensor Network data provided by the Intel lab dataset is not represented directly as a graph, instead the edge probabilities are given to determine the probability with which a sender node can transmit information to a receiver node. This introduced some noise in the data, a simple distribution of probability revealed that many of these edges can be removed without loosing connectivity information. We also verified that the size of Largest Weakly Connected component spanned the entire graph if the Probability was set to $p > 0.4$. The dataset used has sufficient information that can enable usage of Neighbor-Based analysis like DNODA, Community based detection CNA and implement Machine Learning techniques on large time-varying multi-attributed data. This condition is imperative and sufficient for a wireless sensor network where in the average degree is more than one and there is single giant component in the graph.

Standardizing time-varying data: Further we faced the following challenges using the raw sensor data as is:

- Loosely Defined Epoch: Epoch or simply a finite period of time is necessary to establish a baseline while working on time series data similar to the Intel Lab dataset. We found that the concept of epoch was not strongly defined in the dataset, though the readings are supposed to be registered every 31 seconds from each sensor we found that several nodes had missing data for different epochs. In such time-varying data-sets smoothing of data is not a very good idea, since this could be counter-intuitive mainly when the intent is to detect anomalies. We applied a standardizing mechanism and set the epoch size to 60 seconds and mapped latest reading from each sensor across all epochs obtain a dataset of 46,613 epochs * 52 nodes = 2423876 readings.
- Missing Data for Sensors: We found that the connectivity information for MoteIds 0 and 5 were not recorded. These nodes were removed from the dataset.

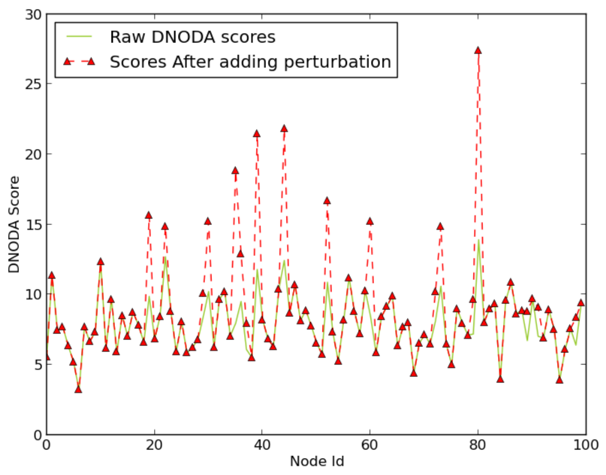
C. Problem statement

Given a graph $G = (V, E, w, D(t))$, where V is a set of n nodes, $E \subseteq V \times V$ is a set of m undirected edges, and $w : V \times V \rightarrow [0, 1]$ is a weight function such as $w_{u,v} = 0$ if and only if $(u, v) \notin E$, $D(t) \in R^d$ is a row vector of time-dependent node attributes, d is number of attributes. With subscript $D_v(t)$ we denote vertex v which has attribute vector $D(t)$ and with superscript $D^i(t)$ we denote i-th component of the node attribute vector $D(t)$, $i \in \overline{1, d}$

Definition (Type 1 Anomaly): For graph G a node v is called type 1 anomaly, if it has attributes $D_v(t)$ that are rare and differ from the majority of other node's attributes $D_u(t)$ for $u \in V \backslash v$, $t \in [t_0, t_0 + \delta)$, for small $\delta > 0$.

This anomaly is referred to as global anomaly, Fig.1.a. This anomaly type does not take into account any network structure in the data. But as we mentioned earlier network data incorporates interdependence, therefore we study two additional pattern-based definitions of anomalies:

Fig. 4: Synthetic Local Anomalies: Spikes in DNODA score



Definition (Type 2 Anomaly): For graph G a node v is called type 2 anomaly, if it has attributes $D_v(t)$ that are rare and differ from the majority of neighboring node's attributes $D_u(t)$, where $u \in N(v)$, where $N(v)$ are neighboring nodes of node v, $t \in [t_0, t_0 + \delta)$, for small $\delta > 0$.

This anomaly is referred to as neighbor anomaly.

Definition (Type 3 Anomaly): For graph G a node v is called type 3 anomaly, if it has attributes$D_v(t)$ that are rare and differ from the majority of the same community node's attributes $D_u(t)$, $u \in C(v)$, where $C(v)$ are nodes from the same community node v, $t \in [t_0, t_0 + \delta)$, for small $\delta > 0$. . By community, we mean a densely connected groups of "close" nodes in the graph. This anomaly is referred to as community anomaly.

D. Algorithms

Direct Neighbour Outlier Detection Algorithm (DNODA) DNODA is an algorithm that considers use of the direct neighbours $u \in N(v)$ of a given node v. A DNODA outlier score is calculated as presented in [9]. Intuitively this score is directly proportional to the distance of v from its direct neighbours using the feature vector $D_v(t)$, refer to score as $X_{DNODA} \in R^d$. Hence an aberrant variation of any node would indicate an anomaly. Refer Algorithm 2 for a simplified overview of the implementation.

Evaluation on the test graph shown in fig 2 clearly shows spikes in the DNODA score for the anomalous nodes. See fig 4

As a quick test on Intel data set, we applied DNODA analysis for one epoch ($\delta = 60 seconds$) with edge probability high enough to ensure a giant connected component of all nodes in the wireless network. The results indicated that for the given time interval MoteId 47 showed abnormal DNODA outlier score hence showing Type 2 Anomaly, refer Fig 5

Algorithm 2 DNODA and CNA

```
procedure $main()$
    $stdEpochData \leftarrow getStandardizedRawData()$
    for $epoch$ in $stdEpochData$ do
        output1 $getDNODAScoreForEpoch(epoch)$
        output2 $getCNAScoreForEpoch(epoch)$

procedure $getDNODAScoreForEpoch(epoch)$
    $distDict \leftarrow$ Dictionary
    for $(minute, moteid)$ in $epoch$ do
        $nbrmoteid \leftarrow getDirectNeighbor(moteid)$
        $dist \leftarrow 0$
        for $nbr$ in $nbrmoteid$ do
            if $nbr$ in $epoch$
                $dist \leftarrow dist+$
                $euclidDistance(epoch[moteid],$
                $epoch[nbr])$
        $distDict[moteid] = dist/|nbrmoteid|$
    return $distDict$

procedure $getCNAScoreForEpoch(epoch)$
    $distDict \leftarrow$ Dictionary
    for $(minute, moteid)$ in $epoch$ do
        $nbrmoteid \leftarrow getCommunityNeighbours(moteid)$
        $dist \leftarrow 0$
        for $nbr$ in $nbrmoteid$ do
            if $nbr$ in $epoch$
                $dist \leftarrow dist+$
                $euclidDistance(epoch[moteid],$
                $epoch[nbr])$
        $distDict[moteid] = dist/|nbrmoteid|$
    return $distDict$
```

Community Neighbor Algorithm (CNA)
Graph partitioning techniques can be used for anomaly detection of large-scale graph data. However partitioning large graphs into meaningful quality partitions is an NP-Complete problem and requires fast and high quality algorithms. Firstly we analyzed the available graph data for existence of Communities, we applied simple visualization on Probability Adjacency Matrix and found that Dense Overlapping and Non-overlapping communities exists in the graph data. refer Fig 6

We used simple, fast and scalable Markov Cluster (MCL) algorithm specified in [17] to partition the graph into meaningful clusters. Intuitively it is expected to have Sensors in the fully connected graph form communities that are closely based on their respective geographical positions. MCL algorithm when applied to the Intel Lab dataset was able to produce Sensor communities which closely followed their respective locations (and hence physical obstacles) in the Intel Lab. See fig 7

When the communities are identified we calculate the score within community as $X_{CNA} = \frac{\sum_{i \in C(t)} D_i(t)}{|C(t)|} \in R^d$. Intuitively this is similar to DNODA but measures distance from its community members vs direct neighbours. Algorithm 2 covers a simplified overview of implementation for this technique.

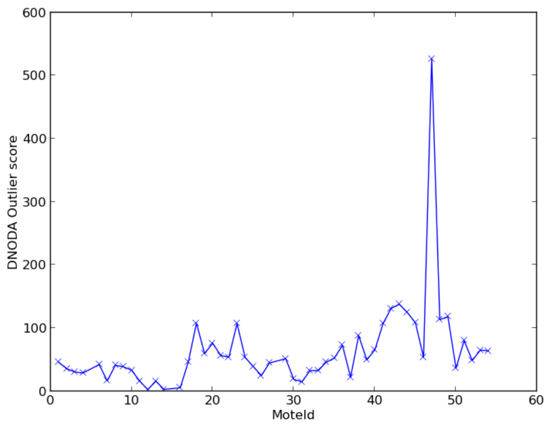


Fig. 5: DNODA score for each MoteId for a specific epoch with $\delta = 60 seconds$ showing Type 2 anomaly at MoteId 47

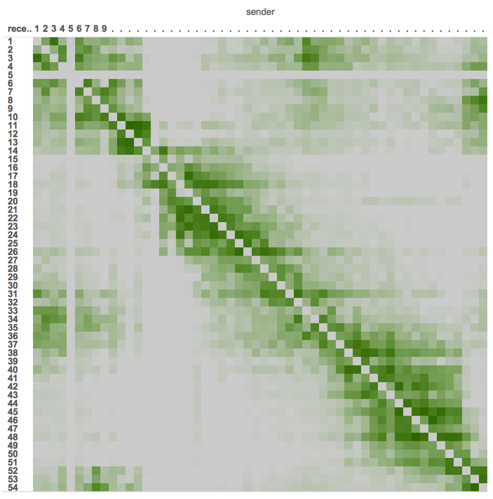


Fig. 6: Visualization of Probability Adjacency Matrix for Existence of Community Structures

CNA technique is particularly useful to identify Type 3 anomalies wherein there are no Local anomalies (Type 2) but outliers exists within the community. This technique when applied on graph from fig 3 we can see the difference between distribution of DNODA scores and corresponding spikes in the CNA scores, refer fig 8

Isolation Forest
The idea of Isolated Forest algorithm is isolation (separating instances from the rest of instances), i.e. use the property of anomalies that they are rare and different. Most anomaly detection methods like one-class SVM [11] try to build a model for ”normal” points, and then find points that do not follow this model. The crucial advantage of Isolation Forest is the speed compared to one-class SVM, and also more complex

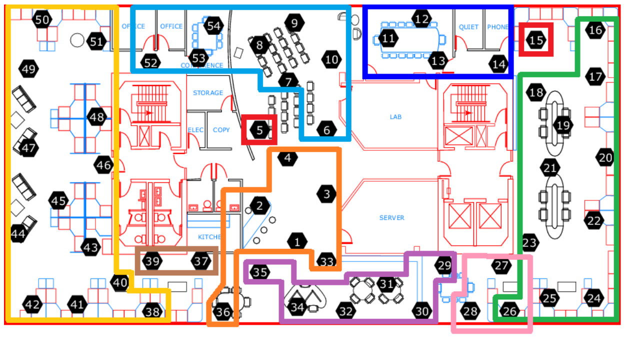


Fig. 7: Sensor Boundaries identified by the Graph communities on Intel Lab dataset [13]. Boundaries closely represent the geographical position of sensor nodes

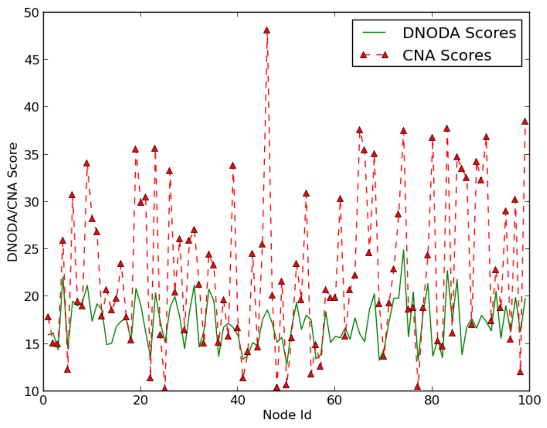


Fig. 8: DNODAScore vs CNA Score on synthetic dataset

decision boundary Fig.10. The main concept of Isolation Forest algorithm is the definition of isolated tree:

Definition (Isolation Tree) [12]: Let T is a node of Isolation Tree. T is either external node with no children, or an internal node with one test condition and two children nodes (T_l, T_r). A test condition is defined by attribute q and a split value p, such that condition $q < p$ splits data points into T_l and T_r

Let's consider an example from original paper [12], Fig. 9.

In this example, partitions are generated by randomly selecting an attribute q and then randomly selecting a split value p between the maximum and minimum values of the selected attribute. For the anomalous point x_0 we need to make only four splits to isolate it from other points using isolated tree, in contrast to completely isolate point x_i we need to do eleven splits. Therefore, intuition is that anomalies are isolated closer to to the root of the tree, whereas normal points are isolated at the deeper end of the tree, and we can use path length within a tree till point is isolated as a anomaly score. Refer to Algorithm 3 for more details. The technique of construction $iForest$ from $iTrees$ using subsampling ψ is commonly used

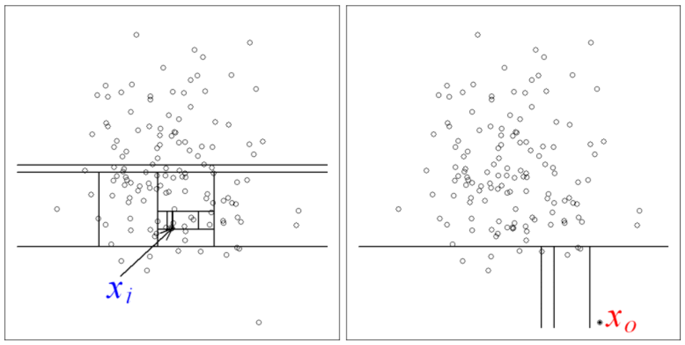


Fig. 9: Example of data split for the case of normal point x_i (left plot), and anomalous case x_0 (right plot)

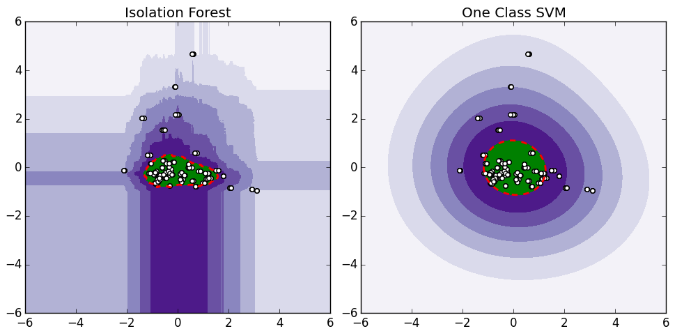


Fig. 10: Comparison of decision boundaries build by Isolation Forest and One-class SVM

in many bagging approaches. As mentioned before, based on split attribute and split values, the dataset is patitioned on X_l and X_r, and so on till point is isolated. Notation for the algorithm, e and $depth_{max}$ -current and maximum tree heights, and scoring function $c(n)$, i.e. average path length $c(n) = 2H(n-1) - (2(n-1)/n)$, where $H(n) = ln(n) + 0.57$ is harmonic number.

For global anomaly detection (type 1 anomalies) we will use feature space which is just attribute information $X = D(t) \in R^d$. For neighbor-based anomaly detection (type 2 anomalies) we will use $X = [D(t), X_{DNODA}] \in R^{2d}$, and for community anomalies detection we will use $X = [D(t), X_{CNA}] \in R^{2d}$

Deep Autoencoders

Deep autoencoders [14] is an unsupervised learning algorithm that is based on neural network and backpropagation, with the response variable equal to the inputs, refer to Fig11.

Intuitively, autoencoder tries to learn an approximation to the identity function, so that the output $\hat{D}$ is similar to the input D. Even though, it seems that identity function can be easily learned, when constraints are imposed on the hidden layer (by number of hidden units) low-dimensional representation of the data can be discovered. Using autoencoders, anomalies can be

Global Anomalies	Local Anomalies	Community Anomalies
iForest_basic 98.1%	DNODA 85.1%	**CNA 90.1%**
Autoencoders 96.4%	**iForest_DNODA 88.6%**	iForest_CNA 76.5%
	Autoencoders_DNODA 82.7%	Autoencoders_CNA 79.9%

TABLE I: Accuracy in terms of area under ROC curve (AUC) for the methods and the three types of anomalies in Synthetic Datasets

Algorithm 3 Isolation Forest

```
procedure $iForest(X, t, \psi)$
    Initialize Forest
    for $i = 1$ to $t$ do
        $X' \leftarrow sample(X, \psi)$
        $Forest \leftarrow Forest \cup iTree(X', 0, depth_{max})$
    return Forest
procedure $iTree(X, e, depth_{max})$
    if $e \geq depth_{max}$ or $|X| \leq 1$
        return exNode$\{Size \leftarrow |X|\}$
    else
        $q = random(1, d)$
        $p = random(\{D_q\}_X)$
        $X_l = filter(X, q < p)$
        $X_r = filter(X, q \geq p)$
        return $inNode\{Left \leftarrow iTree(X_l, e+1, l),$
            $Right \leftarrow iTree(X_r, e+1, l),$
            $SplitAtt \leftarrow, SplitVal \leftarrow p\}$
procedure $AnomScore(x, iTree, e)$
    if $iTree$ is external node: return e + c($iTree$.size)
    $a \leftarrow iTree.splitAtt$
    if $x_a < iTree.splitValue$:
        return $AnomScore(x, iTree.r, e+1)$
    else : return $AnomScore(x, iTree.l, e+1)$
```

detected since they will be different from learned ”normal” patterns, and therefore would have a higher reconstruction error.

More formally, An autoencoder takes input D and first *encodes* it to a hidden representation H through deterministic mapping $H = \sigma(W * D + b)$, for example using sigmoid function σ. Then, hidden representation then mapped back *decoded* into a reconstruction $\hat{D} = \sigma(\tilde{W} * H + \tilde{b})$, where W,$\tilde{W}$,b and $\tilde{b}$ are weights and biases correspondingly. Next, the reconstruction error is calculated as a second norm $L_2(x, z)$. The Hidden units should capture main factors of variation in data.

E. Methods evaluation

1) ***Synthetic Datasets**:* As was mentioned earlier there are no datasets that contains different anomaly types, in particular non-overalping anomaly types. Therefore three synthetic datasets were generated in such a way that they will have only specific type of anomaly, and does not contain any other types of anomalies. This dataset can be downloaded from [15].

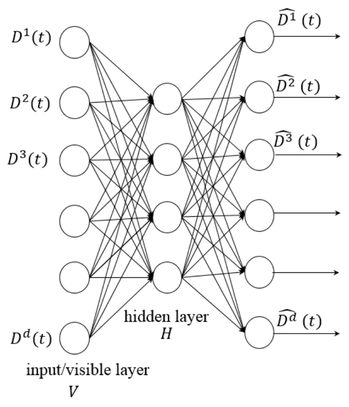


Fig. 11: Deep Autoencoders Architecture. The goal is to obtain in the output values that are close to the input

Most anomaly methods map data to a feature space to emphasise anomalous behaviour, and after that still in feature space you have to make a decision based on a threshold (based on quartiles or changing point methods). The common method for anomaly algorithms benchmarking is area under ROC curve (AUC), basically how good of trade off between precision and recall we can obtain with different thresholds. In **Table I**, AUC for different anomaly types and different methods is presented.

Since iForest algorithm is state of the art for general anomaly detection [16], it outpefroms all even autoencoders for global anomalies. For the case of local anomalies, combination of iForest and DNODA features produces superior performance. As for the community anomalies, simple CNA produces the best result. We believe, that the reason why deep autoencoders showed smaller AUC is twofold. First, the dataset is not big enough for deep learning to outperform other methods. Second, deep autoencoders try to approximate ”normal” datapoints, while iForest tries to isolate specific anomalous point, difference is that non-normal points can be just noise or statistically insignificatnt outliers, while iForest tries to isolate points that represent different behaviour.

2) ***Intel Dataset**:* As a real world application of the proposed anomaly detection methods, we used Intel Dataset, basically to label it with three anomaly types. Intuitively, without graph features, general anomaly detection methods should find global anomalies. Indeed, according to the table below $iForest\_basic$ finds 4398 anomalies, and due to global character of the search all of these anomalies are global (type 1). This method finds extreme outliers - possibly false measurements from the

Time	Node	Temp(T)	Humid(H)	Light(L)	DNODA_T	DNODA_H	DNODA_L	iForest_basic	DNODA_Score	iForest_DNODA
2/28/2004 4:14:00PM	20	-38.4	-4	566.72	23.16	33.41	380.88	**0.81**	447.37	0.38
3/27/2004 1:03:00PM	35	122.15	-3.92	1.38	122.15	-3.92	1,700.16	0.44	**2,406.18**(Fig.12-a)	0.42
3/4/2004 8:25:00PM	8	20.57	39.55	114.08	18.66	44.55	0.46	0.39	120.17	**0.76** (Fig.12-b)

TABLE II: Example of local anomalies in Intel dataset

Method	Anomalies count	% overlay with type 1
iForest_basic	4398 (Type1)	x
DNODA	1403 (Type 2)	9%
iForest_DNODA	3058 (both Type 1 and Type 2)	22%

TABLE III: Intel Dataset Anomalies

sensor, refer to the top of the Table II, $iForest\_basic$ detects anomalous temperature and humidity with iForest anomaly score 0.81.

Next we use DNODA features to use neighboring nodes attributes to find anomaly of type 2. The total number of anomalies discovered is 1403. Intersection of anomalies type 1 and type 2 is 9%. By using iForest over DNODA features both type 1 and type 2 anomalies can be found.

Refer to Fig.12.a, that shows the case of type 2 anomaly. Node 35 has light attribute significantly lower than all its direct neighbors, and as a result it has huge DNODA_score (top table, second line). At the same time both versions of $iForest$ do not capture this anomaly, probably due to existence of other nodes with low light value in the graph at the same epoch.

In the Fig.12.b, node 8, Table II has very low DNODA_score for attribute light, and at this epoch none of the nodes had similar nodes, therefore $iForest\_DNODA$ detects it as anomaly. The application of $iForest\_DNODA$ shows really interesting results, by finding cases that were missed by DNODA. The reason for that is the simplistic thresholding approach that is used in DNODA compare to sophisticated points isolation techniques of $iForest$.

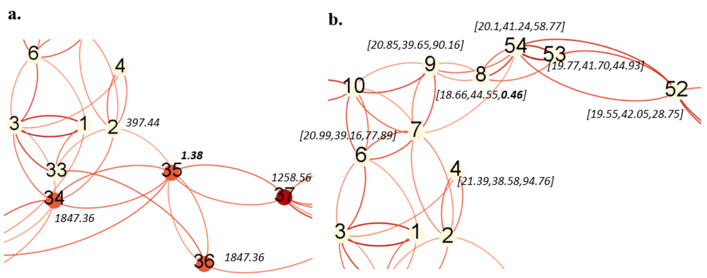


Fig. 12: Example of anomalies detected: a. Type 2 anomaly by basic DNODA algorithm, b. Both Type 1 and Type 2 anomaly by iForest over DNODA features. Text in italics shows the values of node attributes at given epoch: in case of a. it is light attribute, and in case of b. it is DNODA scores [DNODA_T, DNODA_H, DNODA_L]

IV. Conclusion

In this paper we evaluated three types of anomalies, proposed methods to generate synthetic non-overlaping graph anomaly datasets. Several methods were compared using synthetic datasets. For global anomalies, iForest showed the best accuracy. For local anomalies, combination of DNODA and iForest has the best performance, and while for community anomalies CNA showed promissing results. Using these methods, a real-application dataset was labeled.

V. Acknowledgement

Special acknowledgment is made to Paris Syminelakis for the expert guidance and mentoring, help with anomaly definitions and synthetic dataset generation algorithms.

References

[1] Varun Chandola, Anomaly Detection: a survey
[2] Stephen Ranshous, Anomaly detection in dynamic networks: a survey
[3] Leman Akoglu, Graph-based anomaly detection and description: a survey
[4] Yan Yao, Online Anomaly Detection for Sensor Systems: a Simple and Efficient Approach
[5] Ethan Dereszynski, Spatiotemporal Models for Data-Anomaly Detection in Dynamic Environmental Monitoring Campaigns
[6] Misael Mongiovi, Spotting Significant Anomalous Regions on Dynamic Networks
[7] Heng Wang, Locality Statistics for Anomaly Detection in Time Series of Graphs
[8] Leman Akoglu, OddBall: Spotting Anomalies in Weighted Graphs
[9] Jing Gao, On Community Outlier and their Efficient Detection in Information Network $http://hanj.cs.illinois.edu/pdf/kdd10\_jgao.pdf$
[10] George Karypis, A fast and high quality multilevel scheme for partitioning irregular graphs
[11] Larry M. Manevitz, One-Class SVMs for Document Classification
[12] Fei Tiny Liu, Isolation Forest
[13] Intel Lab Dataset, $http://db.csail.mit.edu/labdata/labdata.html$
[14] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle, Greedy layer-wise training of deep networks. In Advances in Neural Information Processing Systems, 2007.
[15] $http://hemalthakkar.com/research/datasets$
[16] A. Emmott, S. Das, T. Dietterich, A. Fern, and W. Wong, Systematic construction of anomaly detection benchmarks from real data, Proc. of the ACM SIGKDD Workshop on Outlier Detection and Description, 2013
[17] MCL a clustering algorithm for graphs $http://micans.org/mcl/index.html?sec_thesisetc$
[18] MM Breunig, HP Kriegel, RT Ng, J Sander, LOF: identifying density-based local outliers,ACM sigmod record, 2000