Local Learning for Mining Outlier Subgraphs from Network Datasets

Manish Gupta Arun Mallya Subhro Roy Jason H. D. Cho Jiawei Han
Microsoft, India University of Illinois at Urbana-Champaign (UIUC), IL, USA

Abstract

In the real world, various systems can be modeled using entity-relationship graphs. Given such a graph, one may be interested in identifying suspicious or anomalous subgraphs. Specifically, a user may want to identify suspicious subgraphs matching a query template. A subgraph can be defined as anomalous based on the connectivity structure within itself as well as with its neighborhood. For example for a co-authorship network, given a subgraph containing three authors, one expects all three authors to be say data mining authors. Also, one expects the neighborhood to mostly consist of data mining authors. But a 3-author clique of data mining authors with all theory authors in the neighborhood clearly seems interesting. Similarly, having one of the authors in the clique as a theory author when all other authors (both in the clique and neighborhood) are data mining authors, is also suspicious. Thus, existence of low-probability links and absence of high-probability links can be a good indicator of subgraph outlierness. The probability of an edge can in turn be modeled based on the weighted similarity between the attribute values of the nodes linked by the edge. We claim that the attribute weights must be learned locally for accurate link existence probability computations. In this paper, we design a system that finds subgraph outliers given a graph and a query by modeling the problem as a linear optimization. Experimental results on several synthetic and real datasets show the effectiveness of the proposed approach in computing interesting outliers.

1 Introduction

With the ever-increasing popularity of entity-centric applications, it has become very essential to study the interactions between entities, which are captured using edges in entity-relationship (or information) graphs. For example, co-authorship graphs capture the co-authorship association between various authors. Similarly, social networks, protein-protein interaction networks, employee networks, etc. also capture a variety of rich associations.

Query Based Subgraph Outliers

Given such an entity-relationship graph, data analysts may be interested in finding anomalous portions of the graph. For example, a security officer may like to find some tiny but suspicious activity clubs from a massive social network, such as Facebook or LinkedIn. Similarly, network security companies might be interested in discovering a group of computers running malicious software as botnets. Besides these, interesting applications of anomalous sub-networks exist in the bio-medical domain too [6]. Often times, a user can express the type of suspicious subgraph she wants to discover in the form of a query. For example, based on the intelligence obtained so far, an analyst would like to gather information about a terrorist ring with particular features. Thus, a query based outlier sub-network detection mechanism becomes a critical piece for any network monitoring system. Given a query, the mechanism is expected to return the matching subgraphs from the original graph ordered by their outlierness score. Instead of taking a general global perspective, the proposed system aims at giving the user a flexibility to find outliers following a particular schema and predicates encoded in the form of a query.

Brief Overview of the Proposed Approach

A subgraph can be considered as an outlier based on its connectivity structure. An outlier subgraph is one which has many unexpected edges and missing many expected edges within itself, and between itself and its neighborhood. Usually the features of two entities in a graph can be used to predict the existence of an edge between these two entities. For example, in a social network, the affiliation, the interests and the age of the person entities are some of the important features that can help predict the probability of a link's existence. The degree to which each of these features determines the link existence probability differs from one part of the network to the other. For example, in some parts of a social network, age is an important factor while for other parts of the network (which deal with people in well known universities), affiliation might be the most important factor. To capture these local correlations of features with link existence, we propose an optimization model which uses the knowledge within a 1-hop neighborhood to infer the feature weights. Besides learning the feature weights, the optimization also helps us to compute the outlier score of a subgraph.

Subgraph Outlier Examples

Subgraph outliers are interesting because they can help identify suspicious or rare subgraphs in the network.

Computer Science Research Example. Consider a co-authorship graph such that two authors are connected if they have collaborated on some paper. Also, assume that we have a community distribution associated with each author which specifies the probability that the author belongs to a research area. Consider a user query containing four author nodes all connected to each other. A normal match to such a query would consist of authors that belong to a single research area with collaborators in the neighborhood belonging to the same area. But a subgraph containing all data mining authors who mostly have theory authors in their neighborhood denotes an interesting region much different from its neighborhood.

Students Network Example. Consider a recently graduated batch of students from some school. Most of them will be connected to others from the same country. Now if say 3 of the students from this batch move to another country for higher studies, (1) most of the edges in their neighborhood will still be between nodes from their origin country, and (2) they will form new friends in their new country and so will have a few inter-country edges. Thus, this group of 3 students will be a subgraph outlier with connectivity very different from their neighborhood.

Comparison with Previous Work

Quite different from existing work which only considers outlier detection of single vertices from networks [7, 9, 10], the proposed work aims at discovering subgraph outliers. Moreover, existing outlier detection work for network data sets has focused on finding outliers for the entire network or in the context of a community. Instead of taking a general global perspective, the proposed system aims at giving the user a flexibility to find outliers following a particular schema and predicates encoded in the form of a query. Compared to our recent work [11, 12] on query based outliers discovered using anomalous associations, the proposed work focuses on query based outliers using neighborhood information. Query based outlier detection on networks can be considered as a special application of graph query processing. Compared to recent work on answering graph queries on networks [21] which provide unranked results, the proposed method attempts to rank such results based on the outlier scores.

Summary

We make the following contributions in this paper.

- We propose and study the problem of identifying subgraph outliers that adhere to an input subgraph query template.
- We propose a methodology to compute the outlierness of a subgraph match based on a max-margin framework. The margin for the linked versus non-linked node pairs in the neighborhood of a subgraph match is used as an indicator of its outlier score.
- Using several synthetic datasets, we compare a local, a partition-wide and a global strategy and show that a local learning strategy provides consistently good accuracy in extracting the injected outliers across a wide variety of experimental settings.
- We also show interesting and meaningful outliers detected from the Four Area and DBLP co-authorship graphs, and the Yeast protein interaction graph.

The paper is organized as follows. In Section 2, we define the subgraph outlier detection problem. In Section 3, we discuss the proposed algorithm to compute the outlier score for a subgraph match. We present results on several synthetic datasets, the Four Area and DBLP co-authorship graphs, and the Yeast protein interaction network with detailed insights in Section 4. We discuss related work and summarize the paper in Sections 5 and 6 respectively. We present interesting discussions related to the proposed approach, subgraph matching algorithm details and details of the real datasets in the supplementary material.

2 Problem Definition

In this section, we formalize the problem definition and present an overview of the proposed system. We start with an introduction to some preliminary concepts.

Definition 1 (An Entity-Relationship Graph). *An entity-relationship graph is an undirected graph $G = \langle V, E, A\rangle$ where V is a finite set of vertices (representing entities) and E is a finite set of edges each being an unordered pair of distinct vertices. Each vertex v has an associated attribute vector which we will denote by $A(v)$. A is a function defined on the vertex set as $A : V \rightarrow [0, 1]^D$ where D is the dimensionality of the attribute vector.*

For this work, we assume numeric attributes. However, the methodology can be used for any type of attributes as long as a dis-similarity function between the two attribute vectors can be defined.

Given a co-authorship graph, each author can be associated with a probability distribution which represents the probability with which the author belongs to a particular research area. Thus the dimensionality is equal to the number of research areas.

Definition 2 (Subgraph Query on a Network). *A subgraph query Q on a network G is a connected subgraph consisting of node set V_Q and edge set E_Q such that $|V_Q| > 1$. The subgraph query Q can be answered by returning all exact matching subgraphs from G.*

Definition 3 (Subgraph Isomorphism). *A graph $g = \langle V_g, E_g\rangle$ is a subgraph of another graph $g' = \langle V_{g'}, E_{g'}\rangle$ if there exists a subgraph isomorphism from g to g'. A subgraph isomorphism is an injective function $F : V_g \rightarrow V_{g'}$ such* *that (1) $\forall v \in V_g$, $F(v) \in V_{g'}$, and (2) $\forall e = (u, v) \in E_g$, $e' = (F(u), F(v)) \in E_{g'}$.*

Definition 4 (Match)**.** *The mapping function F in the subgraph isomorphism definition, with $g = Q$ and $g' = G$ identifies a match for a query Q in G which is an occurrence of Q in G. G may contain multiple such matches of Q. We will denote a particular match by $F = (V_F, E_F)$ where V_F is the set of vertices in F and E_F is the set of edges in F.*

So as to be able to define the outlierness of a subgraph match in terms of node pairs in the neighborhood of a match, we need to consider the induced match rather than the actual match.

Definition 5 (Induced Match)**.** *An induced match $M = (V_M, E_M)$ corresponding to the match F is the subgraph of the graph G induced by the nodes in F. Thus it contains the same nodes as in F but it contains more edges, i.e. $V_M = V_F$. Also, for any pair of vertices u and v of M, (u, v) is an edge of M if and only if (u, v) is an edge of G.*

For simplicity, we will refer to the induced subgraph match as simply "match" in this paper.

Definition 6 (Neighborhood of a Subgraph Match)**.** *Let $N(v)$ denote the vertices and the connecting edges in the neighborhood for a vertex v in graph G, i.e., $N(v)$ contains vertices at hop 1 from v and the edges linking v to those vertices. Let $M = (V_M, E_M)$ be a subgraph match. Thus, $V_M \subseteq V_G$ and $E_M \subseteq E_G$. We define the neighborhood for a match M as the union of the neighborhoods of all its vertices, along with the edges in V_G incident on the vertices in this union. Thus $N(M) = \bigcup_{v \in V_M} N(v) = (V_{N(M)}, E_{N(M)})$. Note that $V_M \subseteq V_{N(M)}$ and $E_M \subseteq E_{N(M)}$.*

Definition 7 (Dis-similarity for a Node Pair)**.** *Each entity in the graph has an attribute vector associated with it. We assume the probability of existence of an edge (and hence the dis-similarity) to be a linear function of the difference in the attribute values of the entities on which the edge is incident. Let w_M denote the feature weight vector for a subgraph M. Also let $|A(v) - A(u)|$ represent the difference vector such that each element is non-negative, i.e., $|A(v) - A(u)|_i = |A(v)_i - A(u)_i|$. Then we define the dis-similarity for the unordered node pair (u, v) with respect to this weight vector as follows.*

$$DisSim(u, v) = w_M^T |A(v) - A(u)| \quad (2.1)$$

Note that we restrict the weight values to be in the range $[0, 1]$. This ensures that the vector w_M can capture the weighted importance of various feature values in determining the dis-similarity between node pairs. Thus $0 \leq w_M(i) \leq 1 \;\; \forall i = 1 \ldots D$. This also ensures that $0 \leq DisSim(u, v) \leq D$ for each node pair (u, v).

We expect the dis-similarity to be high for node pairs which are not linked by an edge, and to be low for node pairs linked by an edge.

Definition 8 (Discriminating Hyperplane)**.** *Given a set of node pairs, some of them may be connected to each other via an edge, while others may not be. Given node pairs (u, v), one can plot them as points in the space of attributes. In such a space, one can learn a weight vector that can act as a separator to discriminate between node pairs connected by edges versus not. Such a weight vector will be referred to as a discriminating hyperplane.*

Definition 9 (Margin)**.** *Consider a set of node pairs (u, v) plotted in the space of attributes. Let H be the maximum dis-similarity for any node pair which is connected by an edge. Let L be the minimum dis-similarity for any node pair which is not connected by an edge. Then, the value $L - H$ is called the margin of separation.*

Definition 10 (Max-Margin Hyperplane)**.** *The discriminating hyperplane w that separates the edge-connected node pairs from the node pairs not linked by an edge with the maximum margin is called as the max-margin hyperplane w^*.*

Definition 11 (Outlier Score of a Subgraph Match)**.** *Let M be a subgraph match. Let S_L be the set of node pairs in the neighborhood of M that are linked by an edge. Let S_{NL} be the set of node pairs in the neighborhood of M that are not linked by an edge. Let w_M^* be the max-margin hyperplane for the points $S_L \cup S_{NL}$ in the attribute space, such that the margin is $L_M - H_M$. We define the outlier score for match M as the negative margin of the max-margin hyperplane.*

$$OS(M) = H_M - L_M \quad (2.2)$$

If the match is not an outlier, the margin between linked node pair instances and non-linked node pair instances will be high. On the other hand, if the match is an outlier, the margin between linked node pair instances and non-linked node pair instances will be low. In fact, when the match is an outlier, the set of linked node pair instances may not even be separable from the non-linked node pair instances, leading to a negative margin value. Thus, this definition of the outlier score of a subgraph match is intuitive.

Note that in the case when the subgraph and its neighborhood is highly homogeneous, i.e., $A(u)$ for all nodes u in the subgraph and its neighborhood is the same, we cannot find a separating hyperplane. In that case, we set the outlier score to 0, i.e., this subgraph is considered extremely normal.

Subgraph Outlier Detection Problem

Given: An entity-relationship graph G, a query Q.

Find: Top few matching subgraphs with highest outlierness scores.

3 Subgraph Outlier Detection

Given an entity-relationship graph G and a query Q, the first step is to identify all the matches of the query Q in the graph G. Let $\mathcal{F}$ be the set of all matches and $\mathcal{M}$ be the set of the corresponding induced matches. Next, an outlier score can be computed for each of these induced matches $M \in \mathcal{M}$. The matches could then be ranked in the non-ascending order of their outlier scores and the top few can be returned as outliers. We use SPath [21] to efficiently compute all the matches for a query Q in graph G. We discuss details of SPath in the supplementary material. We discuss the outlier score computation for every match in Subsection 3.1. This involves estimation of the feature weights for each match M. Based on these, in Subsection 3.2 we present a local subgraph outlier detection algorithm which learns customized feature weights for each match based on the information observed in the local neighborhood of the match.

3.1 Estimating the Weight Vector An SPath-based solution provides us the set of all matches $\mathcal{F}$ for query Q. By simply identifying all the edges in the graph covered by each match, it is straightforward to compute the set of all induced matches $\mathcal{M}$. The next step is to compute the outlier score for each match $M \in \mathcal{M}$. As described in Section 2, the outlier score of a match can be computed in terms of the margin for the max-margin hyperplane, i.e. the best fitting feature weight vector w. In this subsection, we would make some very intuitive observations and encode these observations as an optimization problem which would in turn help us to estimate w, and thereby help in computing the margin.

The max-margin hyperplane should be ideally able to separate the node pairs linked by edges from the node pairs not linked by edges. Also, such a hyperplane should be able to achieve the maximum possible margin for the input dataset. Thus, the objective function to be optimized is as follows.

$$\max \quad L_M - H_M \tag{3.3}$$

Here H_M is the higher bound on the dis-similarity between node pairs linked by edges and L_M is the lower bound on the dis-similarity between the node pairs not linked by edges.

Observation 1. *Consider a node pair (u, v) such that $u \in V_{N(M)}$ and $v \in V_{N(M)}$ and $(u, v) \in E_{N(M)}$, i.e., node u and node v are linked by an edge. Thus, (u, v) is an edge in the local 1-hop neighborhood of the match M. The dis-similarity between the attributes of nodes u and v should not be greater than H_M.*

The above observation can then be expressed in terms of constraints as follows.

For each edge $(u, v) \in E_N(M)$

$$w_M^T |A(u) - A(v)| \leq H_M \tag{3.4}$$

Observation 2. *Consider a node pair (u, v) such that $u \in V_{N(M)}$ and $v \in V_{N(M)}$ and $(u, v) \notin E_{N(M)}$, i.e., node u and node v are not linked by an edge. Thus, (u, v) is a node pair in the local 1-hop neighborhood of the match M but not linked by an edge. The dis-similarity between the attributes of nodes u and v should not be less than L_M.*

The above observation can then be expressed in terms of constraints as follows.

For each node pair (u, v) such that $u \in V_{N(M)}$ and $v \in V_{N(M)}$ but $(u, v) \notin E_N(M)$

$$w_M^T |A(u) - A(v)| \geq L_M \tag{3.5}$$

Observation 3. *w_M represents the weight vector for the max-margin hyperplane to be estimated. The elements of this weight vector need to be bounded and constrained so as to be meaningful. Thus, we constrain the $\{w_M(i)\}_{i=1}^D$'s to be between 0 and 1, and also to sum to 1. The "sum to 1" constraint allows us to interpret w as relative weights assigned to each feature. Constraining the elements between 0 and 1 ensures that the LP does not get biased towards a particular feature (or attribute).*

The above observation can then be expressed in terms of constraints as follows.

$$0 \leq w_M(i) \leq 1 \quad \forall i = 1 \dots D \tag{3.6}$$

$$\sum_{i=1}^{D} w_M(i) = 1 \tag{3.7}$$

A few highly outlier node pairs can lead to zero or negative margin in the non-separable case. To avoid such a situation, we introduce a slack variable for each constraint. Overall, the entire optimization problem with slack variables can be written as follows. C in the objective function is a constant and the second term ensures that the max-margin hyperplane is robust to a few outlier node pairs (i.e., constraints). Recall that S_L is the set of node pairs in the neighborhood of M that are linked by an edge and S_{NL} is the set of node pairs in the neighborhood of M that are not linked by an edge.

$$\max \; L_M - H_M - \frac{C}{|S_L \cup S_{NL}|} \sum_{i=1}^{|S_L \cup S_{NL}|} \xi_i \tag{3.8}$$

subject to the following constraints
For each edge $(u, v) \in E_N(M)$

$$w_M^T |A(u) - A(v)| \leq H_M + \xi_{(u,v)} \tag{3.9}$$

$$\xi_{(u,v)} \geq 0 \tag{3.10}$$

For each node pair (u, v) such that $u \in V_{N(M)}$ and $v \in V_{N(M)}$ but $(u, v) \notin E_N(M)$

$$w_M^T|A(u) - A(v)| \geq L_M - \xi_{(u,v)} \tag{3.11}$$
$$\xi_{(u,v)} \geq 0 \tag{3.12}$$

$$0 \leq w_M(i) \leq 1 \quad \forall i = 1 \ldots D \tag{3.13}$$
$$\sum_{i=1}^{D} w_M(i) = 1 \tag{3.14}$$

The above optimization denoted by the Equations 3.8 to 3.14 is a linear optimization problem which can be solved using the Simplex algorithm. Note that this optimization is very similar in spirit to the soft margin SVM optimization [4].

Values of H_M and L_M

Clearly, if M is an outlier subgraph, i.e., it has unexpected or missing edges within the match or between the match and the neighborhood, then (1) H_M will be very high, and (2) L_M will be very low. In other words, the margin will be very small, indeed it could become negative. Thus, the outlier score, i.e., the negative margin $H_M - L_M$ will be very high. On the other hand, if the match M is not an outlier, (1) H_M will be very low, and (2) L_M will be very high. In other words, the margin will be wide, mostly positive. Thus, the outlier score, i.e., the negative margin $H_M - L_M$ will be very low.

3.2 Subgraph Outlier Detection Algorithm We summarize the overall subgraph outlier detection algorithm in Algorithm 1. For query Q, first all matches $\mathcal{M}$ are computed using SPath (Step 1). Next, for each match M, compute the feature weight vector and the margin (Step 3). The margin is then used to compute the outlier score (Step 4). Finally the matches with very high outlier scores are returned as outliers (Step 6).

Algorithm 1 Subgraph Outlier Detection Algorithm (SODA)

```
Input: (1) Graph $G$, (2) Query $Q$, (3) Parameter $\delta$.
Output: Top subgraph outliers.
1: Compute set of all matches $\mathcal{M}$ for query $Q$ on graph $G$ using SPath($G$, $Q$).
2: for each match $M \in \mathcal{M}$ do
3:     Compute $w_M$ using the optimization specified in Equations 3.8 to 3.14.
4:     Compute outlier score $OS(M)$ using Equation 2.2.
5: Compute the mean $\mu$ and the variance $\sigma^2$ for the outlier scores for all matches.
6: Find subgraph outliers as the subgraphs with outlier score $> \mu + \delta\sigma$.
```

Computational Complexity

Let $|\mathcal{M}|$ be the number of matches, D be the dimensionality of attribute vectors and $|V_Q|$ be the number of nodes in the query. Algorithm 1 is clearly linear in $|\mathcal{M}|$. Let B be the average number of neighbors for any node. Assuming that the neighbor sets for the nodes in the set V_M are disjoint, the maximum number of nodes in the neighborhood of V_M is $B|V_Q|$. So, the maximum number of node pairs in the neighborhood of match M is $(B|V_Q|)^2$. Thus, the optimization consists of $O(2(B|V_Q|)^2 + D + 1)$ constraints and $O((B|V_Q|)^2 + D + 2)$ variables for every match. Solving the optimization is the most expensive step. Linear optimization is polynomial in the number of variables (for interior point methods) or exponential in the number of constraints (for the simplex method) in the worst case. However, in practice simplex takes time linear in the number of constraints for convergence. Also, note that since the computation for each match is independent of those for any other match, these can be done in parallel. In presence of multiple processors, the matches can be distributed evenly among available processors with a shared memory to store the graph.

Choice of Parameters

The proposed algorithm contains two parameters: the outlier deviation δ and the margin versus slack trade-off parameter C in Eq. 3.8. δ is a typical parameter that is used in almost every outlier detection algorithm and has traditionally been set to a value in [1.5,3]. Setting it to a higher value results in higher quality and less number of outliers. Setting it to a lower value results in poor quality and more number of outliers. The parameter C controls the degree to which we want to make the optimization soft. A higher C implies that slack variables can take very small values while a smaller C implies that the constraints can be very soft with high values for slack variables.

4 Experiments

Evaluation of outlier detection algorithms is quite difficult due to the lack of ground truth. We generate multiple synthetic datasets by injecting outliers into normal datasets, and evaluate the outlier detection accuracy of the proposed algorithms on the generated datasets. We also conduct case studies by applying the method to real datasets like DBLP, Yeast-Network and Four Area (a subset of DBLP). We perform comprehensive analysis to justify that the top few outliers returned by the proposed algorithm are meaningful. The data and the code are available at `http://dais.cs.uiuc.edu/manish/LocalLearningOutlier/`.

Baselines

We explore three baselines: Uniform Weight Vector based method, Global Weight Vector based method and Partition-wide Global Weight Vector method.

Uniform Weight Vector

In this approach, rather than estimating w, we fix w to be a constant vector $\overline{1/D}$. This w means that we give equal weight to each of the attributes to determine the dis-similarity between two nodes. No optimization for estimating w is performed in this approach. H_M is computed as the largest dis-similarity value for any node pair under the uniform w and L_M is computed as the smallest dis-similarity value for any node pair under the uniform w. The outlier score is then computed as $H_M - L_M$. We refer to this baseline as *UniformW*.

Global Weight Vector

For this baseline, we randomly choose a set of matches S from the set $\mathcal{M}$. We compute the union set of all nodes in S. Since this set could be quite large, we sample a fixed number of nodes. For these nodes, we compute the edge-linked node pairs and node pairs not linked by edges. Using both these sets of node pairs, we can design a single linear optimization problem. The solution to this optimization problem provides the values for a global w vector. Next, for each match, this w is used to compute H_M as the maximum dis-similarity for any node pair in the neighborhood which is connected by an edge. Similarly, the w is also used to compute L_M as the minimum dis-similarity for any node pair which is not connected by an edge. Finally, $H_M - L_M$ is used as the outlier score for the match. We refer to this baseline as *GlobalW*.

Partition-wide Global Weight Vector

For this baseline, the input graph is split into K partitions using a mincut-based graph partitioning algorithm (METIS [15]). For every partition p, the processing is done independently as follows. Given all the matches for a user query, matches that lie completely within the partition are computed. A match lies completely within the partition iff each of its nodes belong to the vertex set for that partition. From this set of matches, we select a random match. The margin is computed for this match using the same methodology as discussed in Section 3. If the margin is sufficiently high (higher than a threshold), we know that the max-margin hyperplane w reflects the true feature weights in this region. Else, another random match is selected. Once a w with margin greater than the threshold is obtained, this w is then used to compute the outlier score for each match in the partition p. For each match in the partition, this w is used to compute H_M as the maximum dis-similarity for any node pair in the neighborhood which is connected by an edge. Similarly, the w is also used to compute L_M as the minimum dis-similarity for any node pair which is not connected by an edge. Finally, $H_M - L_M$ is used as the outlier score for the match. We refer to this baseline as *PartitionW*.

4.1 Synthetic Datasets Here, we will present the results of extensive experiments on synthetic datasets.

Synthetic Dataset Generation

We generate a variety of synthetic datasets to test multiple settings of number of entities and relationships in the network, degree of outlierness, and the number of attributes associated with the entities. We vary the number of entities (N) as 1000, 2000 and 5000. The number of attributes per entity is varied as 4, 6 and 10. The specific dataset generation details for each such experimental setting are discussed in the supplementary material. Since the aim is to discover query-based outliers, we inject *query-based* outliers themselves. Again, the supplementary material contains outlier injection methodology in detail.

Results on Synthetic Datasets

We run a large number of experiments with a variety of settings: vary number of objects as 1000, 2000 and 5000; vary percentage of outliers as 1%, 2% and 5% (percentage of the number of matches); vary the number of attributes as 4, 6 and 10. Also, we run experiments for the proposed algorithm (*SODA*) as well as for the three baselines: *GlobalW* (GW), *PartitionW* (PW) and *UniformW* (UW). Table 1 shows the results for all the algorithms and all the experimental settings. For each setting, the accuracy value is the precision with which the algorithm is able to extract the injected outliers. Note that since we use the algorithms to extract only as many outliers as we injected, the recall is the same as the precision in this case. The value in each cell of the table is obtained as the average accuracy by running 10 experiments for the corresponding setting. The results show that for all different experimental settings, the proposed algorithm SODA beats the baselines by a significant margin.

Note that as expected the local learning-based proposed algorithm *SODA* has better accuracy compared to PartitionW, which in turn is better than the global-learning based *GlobalW*. SODA is expected to be better than PartitionW because the w is learned at the match level rather than at the partition level. GlobalW uses node pairs from multiple partitions to learn the optimal max-margin hyperplane. Clearly, since the multiple partitions have a different weight vector for attribute importance (i.e., w), therefore, learning a global w leads to relatively inaccurate results. The UniformW baseline does better than the GlobalW baseline. UniformW assumes uniform weighting for all the attributes. Though UniformW may intuitively seem to be a poor baseline, based on the dataset construction method, assigning uniform weights to all attributes is expected to provide fair results. But clearly learning accurate w's using the proposed algorithm *SODA* beats all the baselines.

Also note that the accuracy of the proposed algorithm does not degrade with increase in number of attributes or increase in the size of data. The accuracy remains almost the same even when the degree of outlierness is varied from 1% to 5%. The average accuracy of the four methods across the various experimental settings is as follows.

N	Ψ	$D = 4$				$D = 6$				$D = 10$			
	(%)	SODA	PW	GW	UW	SODA	PW	GW	UW	SODA	PW	GW	UW
1000	1	85.7	**91.1**	12.4	67	**86.2**	77.2	11.1	76.9	**81.4**	80.3	19.5	66.2
	2	**83**	82.3	22.5	71.4	**89.7**	75.4	15.2	73.1	77	**79.2**	27.8	65.5
	5	**81.7**	75.4	23.6	76.8	**92.1**	79.3	29.7	84.6	77.3	**82.8**	31.7	68.9
2000	1	**85**	78	14	80.1	**93.4**	76.1	13.3	79.8	**87.9**	67.6	21.5	69.5
	2	**90.2**	77.1	24.5	79.5	**87.9**	79	31.6	80.5	**92.9**	74.3	29.7	77.1
	5	**91.2**	84.7	36.6	84.7	**93.6**	80.1	40.4	86	**96**	78	45.7	82.9
5000	1	**90**	84.7	21.2	87.7	**85.6**	76.4	19.3	75.3	**89.2**	69.4	28.8	77.7
	2	79.3	**82.7**	40.3	70.5	**90.3**	81	24.3	80	**91.5**	73.9	38.1	79.7
	5	**92.2**	83.7	53.3	86.3	**93.7**	82.7	32.7	84.2	**95**	77.4	52.2	86.9

Table 1: Synthetic Dataset Results (*SODA*=The Proposed Algorithm SODA, *GW*= GlobalW Baseline, *PW*= PartitionW Baseline, *UW*= UniformW Baseline)

SODA: 88.1%, PartitionW: 78.9%, GlobalW: 28.2%, and UniformW: 77.7%.

Table 2 shows the execution time for the algorithms for varying number of nodes and attributes. The experiments were run on a Linux machine with 4 Intel Xeon CPUs with 2.67GHz each. The code was implemented in Java. As expected SODA takes more time compared to PartitionW which is in turn slower than GlobalW. This is because the most expensive step in the computations is solving the linear optimization. SODA needs to solve the linear program (LP) for each match, PartitionW solves a LP for every partition while GlobalW solves only one LP. UniformW takes slightly more time compared to GlobalW but less than PartitionW. Overall, all algorithms are very efficient. Of course, the computations of the LP can be easily parallelized in which case the execution time would depend upon the number of processors available.

N	D	SODA	PW	GlobalW	UniformW
1000	4	1.72	1.29	0.64	0.88
	6	4.89	2.63	1.10	1.63
	10	48.03	24.03	3.79	6.51
2000	4	3.13	2.00	0.98	1.38
	6	6.69	5.37	1.54	2.22
	10	104.55	82.06	11.67	17.57
5000	4	4.14	1.23	1.16	2.30
	6	10.00	4.12	2.69	3.81
	10	106.51	76.59	14.76	23.11

Table 2: Execution Time in seconds (*SODA*=The Proposed Algorithm SODA, *GW*= GlobalW Baseline, *PW*= PartitionW Baseline, *UW*= UniformW Baseline)

	Four Area	DBLP	Yeast Network
Nodes	27199	30599	3112
Edges	66832	146647	12519
Attributes	4	14	400

Table 3: Data Set Details

	Four Area	DBLP	Yeast Network
3 Clique	86390	153336	6590
4 Clique	130389	112851	3134
5 Clique	272900	352389	1937
5 Subgraph	4082687	9472728	264593

Table 4: Number of Subgraph Template Matches

	Four Area	DBLP	Yeast Network
3 Clique	89	385	76
4 Clique	140	265	35
5 Clique	269	796	22
5 Subgraph	4524	23314	3045

Table 5: Execution Time for SODA (in seconds)

4.2 Real Datasets We experiment with three real datasets: Four Area, DBLP and Yeast Network. The details for the datasets are presented in Table 3. Further details about the construction of these datasets can be found in the supplementary material. For experiments on real datasets, we use clique and subgraph queries with 3–5 nodes.

Four Area: This is a co-authorship graph from DBLP for the four areas of data mining (DM), databases (DB), information retrieval (IR) and machine learning (ML) and consists of papers from 20 conferences (5 per area) till 2008. Each author node is associated with the community distribution vector, i.e., the belongingness of the author to each of the four research areas.

DBLP: This is a co-authorship graph based on collaborations in computer science. Each author node is associated with the community distribution vector, i.e., the belongingness of the author to each of the 18 research areas.

Yeast Protein Interaction Network: The Yeast protein interaction network dataset consists of proteins as nodes. Two proteins are connected if they interact with each other. Each node is associated with dipeptide percentage distribution in the FASTA format sequence.

Each dataset has its own unique characteristics. Four Area and yeast networks are small in size, while DBLP is large. Yeast network has large number of attributes compared to the other two. DBLP is much more dense compared to the other two graphs. Thus our choice of datasets covers a good variety of graph characteristics.

Results on Real Datasets

We present a few case studies corresponding to the top outlier discovered by the proposed algorithm for different queries for each dataset. The number of subgraph matches for various queries on the real datasets are shown in Table 4. Table 5 shows the execution time for all the real datasets.

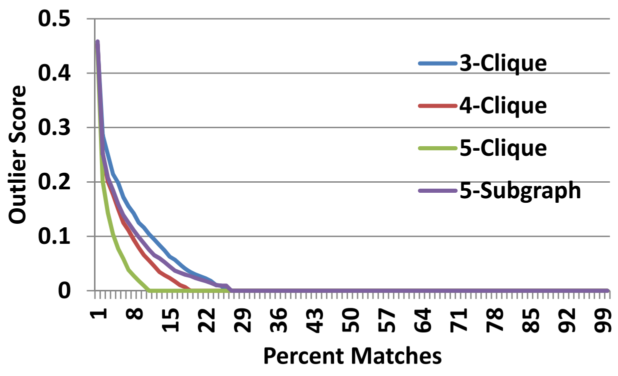

Figure 1: Outlier Score Variation for the Four Area Dataset for four Different Queries

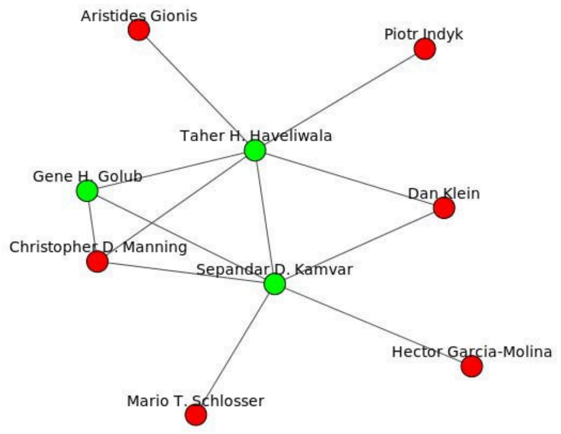

Figure 2: Top Outlier for the 3-Clique Query on Four Area Dataset

Note that the algorithm does not take too long for execution even on very large and dense datasets. In case, the dataset gets more dense, one can easily sample node pairs from the neighborhood, rather than using all the node pairs to compute w. As mentioned before, another option is to perform outlier score computation for every match in parallel to get the results faster.

Figure 1 shows the variation of the outlier score against the percentage of total matches for the Four Area dataset for various queries. Scores below 0 have been set to 0 for good display. Note that for all the queries, a very few matches ($< 0.5\%$) have very high score. Thus, the proposed definition of outlier score clearly possesses the good discriminative ability of an ideal outlier score definition. We observed the same variation for the other real datasets too.

Case Study 1: 3-Clique Query on Four Area Dataset

Top outlier is (Sepandar D. Kamvar, Taher H. Haveliwala, Gene H. Golub). We observed that these authors belong to the IR and ML communities. Also, their neighborhood mainly consists of IR and ML authors. But the outlierness comes in because of anomalous links with some database authors (Hector Garcia-Molina, Piotr Indyk) and also a data mining author (Aristides Gionis). Thus, the set of three authors have inter-disciplinary collaborations which causes them to be marked as an outlier. Figure 2 shows the outlier subgraph (green nodes) with its neighborhood (red nodes).

Case Study 2: 5-Subgraph Query on DBLP Dataset

This query contains 5 nodes and 8 out of the 10 edges. Top outlier is (Paul Holleis, Seong Bae Park, William D Smart, Paul E Dunne, Milde M S Lira) with the following absent edges (Paul E Dunne, Seong Bae Park) and (Paul E Dunne, William D Smart). Note that in this subgraph, except Paul Holleis, all other authors work in Artificial Intelligence. On the other hand, Paul works on HCI, Networks and Algorithms. Thus, especially the edges (Paul Holleis, Paul E Dunne) and (Paul Holleis, William D Smart) are highly abnormal. Further, we analyzed the authors who collaborate with this set of authors. While a majority of these authors focus mainly on Artificial Intelligence, a large percent of their collaborators focus on multimedia and networks. Thus, the AI core of these four authors (Seong Bae Park, William D Smart, Paul E Dunne, Milde M S Lira) seems to be embedded in a region of multimedia and networks folks. Hence, this case is an interesting outlier.

Case Study 3: 4-Clique Query on Yeast Network

Top outlier is (ydl147w, ydr394w, ydr427w, yfr010w). These four proteins and other interacting proteins contain a large percentage of the following dipeptides: LK, LL, EL, LS, LE, SL, SS, AL, EE, KL, LA, EK, DL, KE, VL, IL, AA, LI, DE, IS. However, a few proteins (like ydr201w, yhr027c, yfr052w, ynl250w, ydl147w, ymr308c, ylr106c) contain very small amounts of these dipeptides. Instead their sequences contain high percentages of other dipeptides like IE, LD, KK, KS, LN, NL, AS, DA, EN, LQ. This results in a very small margin because of these few outlier proteins leading to a high outlier score.

5 Related Work

The proposed work is most related to the area of outlier detection for graph data. Work has been done in the data mining community in this area for both static and dynamic graphs.

Static Networks: For static networks, outlier detection has been performed using a variety of techniques discussed in brief as follows. (1) MDL: The minimum description length (MDL) principle can help to identify frequent subgraph patterns and hence outliers in graphs. Noble and Cook [17] propose size of substructure multiplied by its frequency as a measure of its outlierness. Chakrabarti [3] uses the MDL principle to spot anomalous edges. (2) Egonets: In [2], several new rules (power laws) in density, weights, ranks and eigenvalues that seem to govern the "neighborhood subgraphs" (egonets) are discovered and distance to the fitting line representing the power laws is used as the measure for outlier detection. MetricForensics [13] uses a collection of global, local and community level graph metrics to find four novel phenomena in such graphs: elbows, broken correlations, prolonged spikes, and lightweight stars. (3) Random walks: In [16, 20] proximity and random walks are used to assess the normality of nodes in bipartite graphs. (4) Random Field Models: Using a heterogeneous random field model, Qi et al. [19] discover outlier links in a network of users, media objects and tags. Gao et al. [7] perform community detection on networks and outlier detection together in an integrated framework using Hidden Markov Random Fields (HMRFs) to identify community outliers.

Temporal Networks: For temporal graphs, there are three main lines of work [8]. (1) Graph Similarity based Outlier Detection Algorithms: Various graph distance metrics can be used to create time series of network changes by sequentially comparing graphs from adjacent periods. These time series are individually modeled as univariate autoregressive moving average (ARMA) processes and then outliers can be detected [5, 18]. (2) Evolutionary Community Outlier Detection Algorithms: Gupta et al. [9, 10] study an interplay of community detection and temporal outlier detection to discover evolutionary community outliers and community trend outliers. (3) Online Graph Outlier Detection Algorithms: These include identifying anomalous graph linkage structures [1] from a stream of graphs using reservoir sampling. Spectral methods [14] have been also been used for the same.

The proposed work is for static networks. Work on identifying outliers from egonets is the closest to our work. But egonets based work focused on finding outliers in connectivity structure while we focus on connectivity as well as attribute values to compute neighborhood-sensitive outliers. Also, the proposed work follows a query-based paradigm giving the user a higher level of flexibility in expressing her need to find outliers following a particular subgraph schema.

6 Conclusion

We proposed the problem of identifying subgraph outliers that adhere to an input subgraph query template based on deviations in linkage compared to the neighborhood. We discussed a methodology to compute the outlierness of a subgraph match based on a max-margin framework. Using several synthetic datasets, we observed that a local method outperforms a partition-wide approach which in turn is more accurate than a global strategy in extracting the injected outliers across a wide variety of experimental settings. We also showed interesting and meaningful outliers detected from the Four Area and DBLP co-authorship graphs, and the Yeast protein interaction graph. In the future, we would like to perform neighborhood-based subgraph outlier query answering for temporal graphs. Studying the problem for very high dimensional data could also be an interesting direction.

7 Acknowledgements

The work was supported in part by the U.S. Army Research Laboratory under Cooperative Agreement No. W911NF-11-2-0086 (Cyber-Security) and W911NF-09-2-0053 (NS-CTA), the U.S. Army Research Office under Cooperative Agreement No. W911NF-13-1-0193, and U.S. National Science Foundation grants CNS-0931975, IIS-1017362, and IIS-1320617.

We would also like to thank the Institute for Genomic Biology at University of Illinois, Urbana Champaign for their equipment.

References

[1] C. C. Aggarwal, Y. Zhao, and P. S. Yu. Outlier Detection in Graph Streams. In *ICDE*, pages 399–409, 2011.

[2] L. Akoglu, M. McGlohon, and C. Faloutsos. Oddball: Spotting anomalies in weighted graphs. In *PAKDD*, pages 410–421, 2010.

[3] D. Chakrabarti. AutoPart: Parameter-free Graph Partitioning and Outlier Detection. In *PKDD*, pages 112–124, 2004.

[4] C. Cortes and V. Vapnik. Support-Vector Networks. *Machine Learning*, 20(3):273–297, 1995.

[5] P. Dickinson and M. Kraetzl. Novel Approaches in Modelling Dynamics of Networked Surveillance Environment. In *Fusion*, volume 1, pages 302–309, 2003.

[6] Y. Feng, J. A. Syrkin-Nikolau, and E. S. Wurtele. Creating Subnetworks from Transcriptomic Data on Central Nervous System Diseases informed by a Massive Transcriptomic Network. *Interdisciplinary Bio Central (IBC)*, 5(1):1–8, Jan 2013.

[7] J. Gao, F. Liang, W. Fan, C. Wang, Y. Sun, and J. Han. On Community Outliers and their Efficient Detection in Information Networks. In *KDD*, pages 813–822, 2010.

[8] M. Gupta, J. Gao, C. Aggarwal, and J. Han. Outlier Detection for Temporal Data. *TKDE*, 2014.

[9] M. Gupta, J. Gao, Y. Sun, and J. Han. Community Trend Outlier Detection using Soft Temporal Pattern Mining. In *ECML PKDD*, pages 692–708, 2012.

[10] M. Gupta, J. Gao, Y. Sun, and J. Han. Integrating Community Matching and Outlier Detection for Mining Evolutionary Community Outliers. In *KDD*, pages 859–867, 2012.

[11] M. Gupta, J. Gao, X. Yan, H. Cam, and J. Han. On Detecting Association-Based Clique Outliers in Heterogeneous Information Networks. In *ASONAM*, 2013.

[12] M. Gupta, J. Gao, X. Yan, H. Cam, and J. Han. Top-K Interesting Subgraph Discovery in Information Networks. In *ICDE*, 2014.

[13] K. Henderson, T. Eliassi-Rad, C. Faloutsos, L. Akoglu, L. Li, K. Maruhashi, B. A. Prakash, and H. Tong. Metric Forensics: A Multi-level Approach for Mining Volatile Graphs. In *KDD*, pages 163–172, 2010.

[14] T. Idé and H. Kashima. Eigenspace-based Anomaly Detection in Computer Systems. In *KDD*, pages 440–449, 2004.

[15] G. Karypis and V. Kumar. A Fast and High Quality Multilevel Scheme for Partitioning Irregular Graphs. *J. on Sci. Computing*, 20(1):359–392, Dec 1998.

[16] H. D. K. Moonesignhe and P.-N. Tan. Outlier Detection Using Random Walks. In *ICTAI*, pages 532–539, 2006.

[17] C. C. Noble and D. J. Cook. Graph-Based Anomaly Detection. In *KDD*, pages 631–636, 2003.

[18] P. Papadimitriou, A. Dasdan, and H. Garcia-Molina. Web Graph Similarity for Anomaly Detection. *J. Internet Services and Apps*, 1(1):19–30, 2010.

[19] G.-J. Qi, C. C. Aggarwal, and T. S. Huang. On Clustering Heterogeneous Social Media Objects with Outlier Links. In *WSDM*, pages 553–562, 2012.

[20] J. Sun, H. Qu, D. Chakrabarti, and C. Faloutsos. Neighborhood Formation and Anomaly Detection in Bipartite Graphs. In *ICDM*, pages 418–425, 2005.

[21] P. Zhao and J. Han. On Graph Query Optimization in Large Networks. *PVLDB*, 3(1):340–351, 2010.