Focused Clustering and Outlier Detection in Large Attributed Graphs

Bryan Perozzi[◇], Leman Akoglu[◇]
[◇] Stony Brook University
Department of Computer Science
{bperozzi, leman}@cs.stonybrook.edu

Patricia Iglesias Sánchez[○], Emmanuel Müller[○•]
[○] Karlsruhe Institute of Technology, [•] University of Antwerp
Department of Computer Science
{patricia.iglesias, emmanuel.mueller}@kit.edu

ABSTRACT

Graph clustering and graph outlier detection have been studied extensively on plain graphs, with various applications. Recently, algorithms have been extended to graphs with attributes as often observed in the real-world. However, all of these techniques fail to incorporate the user preference into graph mining, and thus, lack the ability to steer algorithms to more interesting parts of the attributed graph.

In this work, we overcome this limitation and introduce a novel user-oriented approach for mining attributed graphs. The key aspect of our approach is to infer user preference by the so-called focus attributes through a set of user-provided exemplar nodes. In this new problem setting, clusters and outliers are then simultaneously mined according to this user preference. Specifically, our FocusCO algorithm identifies the focus, extracts focused clusters and detects outliers. Moreover, FocusCO scales well with graph size, since we perform a local clustering of interest to the user rather than global partitioning of the entire graph. We show the effectiveness and scalability of our method on synthetic and real-world graphs, as compared to both existing graph clustering and outlier detection approaches.

Categories and Subject Descriptors

H.2.8 [**Database Applications**]: Data mining; I.5.3 [**Pattern Recognition**]: Clustering

Keywords

focused graph mining; infer user preference; attributed graphs; clustering; outlier mining; distance metric learning

1. INTRODUCTION

Many real-world graphs have attributes associated with the nodes, in addition to their connectivity information. For example, social networks contain both the friendship relations as well as user attributes such as interests and demographics. A protein-protein interaction network may not only have the interaction relations but the gene expressions associated with the proteins. Both types of information can be described by a graph in which nodes represent the objects, edges represent the relations between them, and feature vectors associated with the nodes represent the attributes. Such graph data is often referred to as an *attributed graph*.

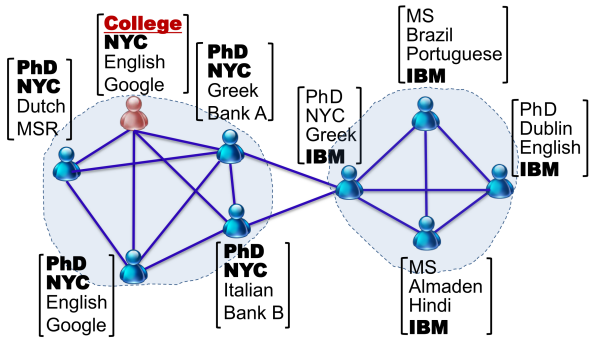


Figure 1: Example graph with two focused clusters and one focused outlier.

For attributed graphs, we see major challenges that remain unsolved by traditional graph mining techniques [4, 10, 19, 24, 31], which consider plain graphs (without attributes). Recent methods have been proposed for attributed graphs, however, they either use all the given attributes [2, 14, 20, 34] or they perform an unsupervised feature selection [16, 18, 22, 26]. In contrast to all of these graph mining paradigms (cf. Table 1), we consider a user-oriented setting where the users can control the relevance of attributes and as a consequence, the graph mining results.

In particular, we consider cluster and outlier detection based on user preference. This *focused clustering* is of particular interest in attributed graphs, where users might not be concerned with all but a few available attributes. As different attributes induce different clusterings of the graph, the user should be able to steer the clustering accordingly. As such, the user controls the clustering by providing a set of exemplar nodes (perceived similar by the user) from which we infer *attribute weights* of relevance that capture the user-perceived similarity. The essence of user preference is captured by those attributes with large weights. We call these the *focus attributes*, which form the basis of our approach for discovering focused clusters and outliers.

To elaborate on this new terminology, we give a toy example in Figure 1. The graph represents the friendship relations and the node attributes denote `degree, location, mother tongue`, and `work`. There are two focused clusters: On the

KDD'14, August 24–27, 2014, New York, NY, USA.

http://dx.doi.org/10.1145/2623330.2623682

Table 1: Comparison of related work.

Property	Graph clustering	On attributed graphs	Attribute subspace	User-preferred clusters	Overlapping clusters	Outlier detection	Scalability
METIS [19], Spectral [24], Co-clustering [10]	✓						✓
Autopart, Cross-associations [7]	✓					✓	✓
PageRank-Nibble [3], [30], BigClam [33]	✓				✓		✓
Spectral Relational [20], SA-Cluster [34]	✓	✓					
CoPaM [22], Gamer [16]	✓	✓	✓				✓
PICS [2]	✓	✓					✓
CODA [14]	✓	✓				✓	
GOutRank [23], ConSub [18]		✓	✓			✓	✓
FocusCO [this paper]	✓	✓	✓	✓	✓	✓	✓

left, people know each other due to their `degrees` and `locations`. On the right, a similarity in `work` induces the second cluster. As such, different user interest in subsets of the attributes may induce different clusters. In case a user is interested in `degree` and `location`, focused clustering should only find the left cluster and not the right one. Analogously, the example outlier is deviating with a college degree among all others having PhDs, where `degree` is a focus attribute.

While our example is on a toy graph, our problem setting has several practical applications in the real-world. For instance, a marketing manager interested in selling cosmetics could aim to find communities in a large social network with its members being of a certain age, gender, and income-level. S/he could then offer deals to a few members from each community, and expect the news to propagate by the word-of-mouth. A scientist could aim to identify clusters of sky-objects that are all in close-distance to one another (assuming a graph is constructed among sky-objects by distance in space) and share certain characteristics of interest (e.g., helium level, temperature, light, etc.).

Such user-preferred clusters are likely a handful in the graph, and thus an algorithm should be able to (*i*) effectively identify user preference, (*ii*) efficiently "chop out" relevant clusters locally without necessarily partitioning the whole graph, and additionally (*iii*) spot outliers if any. In this paper, we offer the following contributions:

- **Focused Graph Mining**: We propose a new user-centric problem setting that exploits the user interest for focused mining in attributed graphs.
- **Steered Search by User Preference**: We infer user preference and steer the graph clustering and outlier detection accordingly.
- **Scaling to Large Graphs:** Our proposed method has near-linear time complexity, and with appropriate initialization can run in time sub-linear w.r.t. the size of the input graph.

In our evaluation we demonstrate the effectiveness and scalability of our method on synthetic and real-world graphs, compared to existing graph clustering and outlier detection methods. Our experiments show that existing approaches are not suitable for the new focused graph mining setting.

2. RELATED WORK

We show the highlights of related work in Table 1. The two key differences of our work are summarized as follows: (1) we introduce a new user-oriented problem setting for attributed graphs, in which we aim to find *focused* clusters and outliers based on *user preference*, and (2) we propose an algorithm that *simultaneously* extracts relevant clusters and outliers from large graphs. In the following, we discuss related work in three areas; traditional plain graph mining, attributed graph mining, and semi-supervised data mining.

Graph mining on plain graphs.

Graph partitioning has been well studied in the literature. Widely used methods include METIS [19] and spectral clustering [11, 24], which aim to find a k-way partitioning of the graph. Different from partitioning, community detection methods [12] cluster the graph into variable size communities. Autopart, cross-associations [7], and information-theoretic co-clustering [10] are parameter-free examples to graph clustering methods. Several methods [3, 30, 33] also allow clusters to overlap as observed in real-world social and communication networks. Works that aim to spot structural outliers in plain graphs include [1, 28]. However, all of these methods are limited to plain graphs (without attributes).

Graph mining on attributed graphs.

Compared to the wide range of work on plain graph mining, there has been much less work on attributed graphs. The representative methods [2, 14, 17, 20, 27, 34] aim to partition the given graph into structurally dense and attribute-wise homogeneous clusters, detect deviations from frequent subgraphs [25], or search for community outliers in attributed graphs [14]. These methods, however, enforce attribute homogeneity in all attributes. Recently some methods loosen this constraint by unsupervised feature selection [26], subspace clustering [16, 22] and subspace outlier detection [18, 23] and extract cohesive subgraphs with homogeneity in a subset of attributes. However, all of these methods either do not perform a selection of attributes or do not allow for user preference to steer the algorithm.

Semi-supervised methods.

A broad variety of methods for semi-supervised clustering consider user-given constraints like 'must-link' and 'cannot-link' referred to as constraint-based clustering [5]. There also exist methods for semi-supervised outlier mining [13]. However, these methods are based on vector data and further, not applicable to graphs with attributes.

Methods on seeded community mining [3, 8, 15, 30] find communities around (user-given) seed nodes. However, those

methods find structural communities on plain graphs and neither apply to attributed graphs, nor enable user preference on attributes. Moreover, they do not provide outlier detection. In contrast, we use user-given exemplar nodes to automatically infer user preference on attributes. To the best of our knowledge this problem setting is new and we propose the first focused graph mining approach for simultaneous clustering and outlier detection in attributed graphs.

3. METHOD FocusCO

In this section we first introduce the notation and pose the focused clustering and outlier detection problem formally. Next, we discuss the main components of our approach and walk through the details of our algorithm. Lastly, we analyze the computational complexity of FocusCO.

3.1 Problem Formulation

In this paper we introduce the novel problem of *focused clustering and outlier detection* in attributed graphs, defined as follows: Given a large attributed graph $G(V, E, F)$ with $|V| = n$ nodes and $|E| = m$ edges, where each node is associated with $|F| = d$ attributes (features), extract from G only the (type of) clusters pertaining to a user u's interest (rather than partitioning the whole graph). To do so, the user provides a small set C_{ex} of exemplar nodes that s/he considers to be similar to the type of nodes the clusters of his/her interest should contain. Assuming that the nodes in a cluster "unite" or "knit up" around a few defining attributes, we then aim to infer the implicit weights $\boldsymbol{\beta_u}$ (i.e., relevance) of attributes that "define" the nodes in C_{ex}, i.e., the weights of attributes that make them as similar as possible. Thus, $\boldsymbol{\beta_u}$ is expected to be a sparse vector with large weights for only a few attributes (e.g., `degree` and `location` in Figure 1), which we call the *focus attributes*.

Having inferred the attribute weights $\boldsymbol{\beta_u}$ from user u, our first goal is to extract focused clusters $\mathcal{C}$ from G that are (1) structurally dense and well separated from the rest of the graph, as well as (2) consistent on the focus attributes with large weights. The focused clusters can be overlapping, sharing several of their nodes, as observed in real-world social and communication networks. Moreover, the set $\mathcal{C}$ is a subset of all the clusters in G since different sets of clusters are expected to unite around different attributes and we aim to extract only those that are specifically similar to the type of clusters user u is interested in. Besides focused clustering, our second goal is to also perform *outlier detection*. Outliers $\mathcal{O}$ are those nodes that structurally belong to a focused cluster (i.e., have many cluster neighbors), but deviate from its members in some focus attributes. In summary, the focused clustering and outlier detection problem in attributed graphs is given as follows:

Given a large graph $G(V, E, F)$ with node attributes, and a set of exemplar nodes C_{ex} of user u's interest;
Infer attribute *weights* $\boldsymbol{\beta_u}$ of relevance/importance,
Extract *focused* clusters $\mathcal{C}$ that are (1) dense in graph structure, and (2) coherent in heavy focus attributes,
Detect focused outliers $\mathcal{O}$, i.e. nodes that deviate from their cluster members in some focus attributes.

3.2 Approach and Algorithm Details

Next we present the details of the three main components of FocusCO: (1) inferring attribute weights, (2) extracting focused clusters, and (3) outlier detection.

3.2.1 Inferring Attribute Relevance

Our focused clustering setting is a user-oriented one, where each user is interested in extracting certain kind of clusters from a given graph. The user steers the clustering by providing a small set of exemplar nodes that are similar to one another as well as similar to the type of nodes the clusters of his/her interest should contain. Our first goal then is to identify the relevance weights of node attributes that make the exemplar nodes similar to each other. This kind of weighted similarity is often captured by the (inverse) Mahalanobis distance: the distance between two nodes with feature vectors $\boldsymbol{f_i}$ and $\boldsymbol{f_j}$ is $(\boldsymbol{f_i} - \boldsymbol{f_j})^T \boldsymbol{A} (\boldsymbol{f_i} - \boldsymbol{f_j})$. Setting $\boldsymbol{A}$ as the identity matrix yields Euclidean distance, otherwise the features/dimensions are weighted accordingly.

Given the exemplar nodes, how can we learn an $\boldsymbol{A}$ such that they end up having small distance to each other? This is known as the distance metric learning problem [29]. We adopt the optimization objective by [32]:

$$\min_{\boldsymbol{A}} \sum_{(i,j)\in P_S} (\boldsymbol{f_i} - \boldsymbol{f_j})^T \boldsymbol{A} (\boldsymbol{f_i} - \boldsymbol{f_j}) \quad (1)$$
$$- \gamma \log \left(\sum_{(i,j)\in P_D} \sqrt{(\boldsymbol{f_i} - \boldsymbol{f_j})^T \boldsymbol{A} (\boldsymbol{f_i} - \boldsymbol{f_j})} \right)$$

which is convex and enables efficient, local-minima-free algorithms to solve it, especially for a diagonal solution.

We give the details of inferring attribute weights in Procedure P1. P_S and P_D are two sets of similar and dissimilar pairs of nodes, respectively (P1 Line 1). In our setting, all pairs of exemplar nodes constitute P_S (P1 Line 2). We create P_D by randomly drawing pairs of nodes that do not belong to the exemplar set (P1 Lines 3-7).

We remark that in creating P_D, we may also obtain samples similar to those in P_S since these draws are random. To alleviate the affect of such draws, we keep the size of P_D sufficiently large. This assumes that the number of dissimilar pairs is substantially larger than the number of similar pairs in the original distribution. This a suitable assumption, given that the number of "focused" clusters for any particular user-preference is likely small. Thus, we make the size of P_D be $|F|$ times larger than that of P_S (P1 Line 7). This also ensures that the data size exceeds dimension size, and that the learning task is feasible.

Moreover, in inferring attribute weights we learn a diagonal $\boldsymbol{A}$ matrix (P1 Line 9). The reason for this choice is twofold. First, individual weights for attributes provide ease of interpretation. Second, learning a diagonal $\boldsymbol{A}$ is computationally much more tractable (especially in high dimensions) than learning a full one, since the latter requires solving a program with a semi-definite constraint. Of course if desired, one can instead learn a full matrix (in low dimensions).

3.2.2 Focused Cluster Extraction

Having determined attribute weights $\boldsymbol{\beta}$, we extract the focused clusters of interest. The main idea in "chopping out" focused clusters from G is to first identify good candidate nodes that potentially belong to such clusters, and then to expand around those candidate nodes to find the clusters. Details are given in Algorithm A1, which we describe below.

The process of finding good candidate sets to expand is detailed in Procedure P2. Intuitively, nodes in focused clusters have high weighted similarity to their neighbors. Therefore, we first re-weigh the edges E by the weighted similarity of

Procedure P1 InferAttributeWeights

Input: exemplar set of nodes C_{ex}
Output: attribute weights vector $\boldsymbol{\beta}$

// generate similar and dissimilar node pairs
Similar pairs $P_S = \emptyset$, Dissimilar pairs $P_D = \emptyset$
for $u \in C_{ex}, v \in C_{ex}$ **do** $P_S = P_S \cup (u, v)$ **end for**
repeat
  Random sample u from set $V \backslash C_{ex}$
  Random sample v from set $V \backslash C_{ex}$
  $P_D = P_D \cup (u, v)$
until $d|P_S|$ dissimilar pairs are generated, $d = |F|$
Oversample from P_S such that $P_S = P_D$
Solve objective function in Equ. (1) for diagonal $\boldsymbol{A}$
return $\boldsymbol{\beta} = diag(\boldsymbol{A})$

their end nodes (P2 Lines 2-4), induce G on the edges with notably large weights[1] (P2 Line 5), and consider the nodes in the resulting connected components as our candidate nodes (P2 Lines 6-7). We call each such component a *core* set.

Next, we expand around each core by carefully choosing new nodes to include in the cluster and continue expanding until there exist no more nodes that increase the quality of the cluster. There exist several measures of cluster quality including modularity and cut size [24]. In this work, we use conductance [3] as it accounts for both the cut size as well as the total volume/density retained within the cluster. The weighted conductance $\phi^{(w)}(C, G)$ of a set of nodes $C \subset V$ in graph $G(V, E, F)$ is defined as

$$\phi^{(w)}(C, G) = \frac{W_{cut}(C)}{WVol(C)} = \frac{\sum_{(i,j)\in E, i\in C, j\in V\setminus C} w(i,j)}{\sum_{i\in C}\sum_{j,(i,j)\in E} w(i,j)}$$

where $WVol(C)$ is the total weighted degree of nodes in C. The lower the conductance of a cluster, the better its quality is with few cross-cut edges and large within-density.

The expansion operation is presented in Procedure P3. First, we enlist all their non-member neighbors as the candidate set (P3 Line 4). For each candidate node n, we compute the difference $\Delta\phi_n^{(w)}$ in cluster conductance if n was to be added to C (P3 Lines 6-16). If there exist any node with negative Δ (i.e., node improves conductance), we pick the best n with the minimum (i.e., largest absolute drop in conductance) (P3 Lines 17-23). We continue iterating until no candidate node yields negative $\Delta\phi^{(w)}$.

The node additions are based on a best-improving search strategy. While being cautious in which node to add (the best one at every step), our decisions are greedy. Thus, in the following step of our algorithm we adopt a retrospective strategy and check if there exist any nodes in C whose removal would drop conductance, presented in Procedure P4. We repeat the node addition and removal iterations until convergence, that is, when the conductance stops changing (A1 Lines 7-12).

We remark that our algorithm is guaranteed to converge; as the (weighted) conductance of a cluster is lower-bounded by 0 and we improve (i.e., decrease) the weighted conductance in every iteration (P3 Line 8, P4 Line 5).[2]

We omit the details of the Δ conductance computation for brevity, but remark that it is an efficient operation. Specifically, the operation of a node u to be added to or to be removed from a cluster S has complexity proportional to the degree of u, i.e. $O(d(u))$. In addition, the total volume of S is simply increased/decreased by the weighted degree $w(u)$ of u, when it is added/removed, which takes $O(1)$.

[1]To identify such edges, we use hypothesis testing. We first find the top few most weighted edges, and bootstrap a Normal distribution. We then progressively subject the remaining edges to a membership-test and consider only those that pass the test. Every time an edge passes the test, model parameters are updated.

Algorithm A1 FocusCO: Focused Clusters&Outliers

Input: attributed graph $G(V, E, F)$, exemplar nodes C_{ex}
Output: focused clusters $\mathcal{C}$ and outliers $\mathcal{O}$

$Cores \leftarrow$ FindCoreSets$(G(V, E, F), C_{ex})$
$\mathcal{C} = \emptyset$, $\mathcal{O} = \emptyset$
for each core $i \in Cores$ **do**
  $C \leftarrow Seeds(i).getNodes()$
  $BSN = \emptyset$ *// holds all Best Structural Nodes to add*
  $\phi_{curr}^{(w)} \leftarrow \phi^{(w)}(C, G)$
  repeat
    $\phi_{init}^{(w)} \leftarrow \phi_{curr}^{(w)}$
    $(C, BSN, \phi_{curr}^{(w)}) \leftarrow$ Expand$(G, C, BSN, \phi_{curr}^{(w)})$
    $(C, \phi_{curr}^{(w)}) \leftarrow$ Contract$(G, C, \phi_{curr}^{(w)})$
    $BSN \leftarrow BSN \backslash C$
  until $\phi_{init}^{(w)} = \phi_{curr}^{(w)}$
  $\mathcal{C} \leftarrow \mathcal{C} \cup C$, $\mathcal{O} \leftarrow \mathcal{O} \cup BSN$
end for

Procedure P2 FindCoreSets

Input: attributed graph $G(V, E, F)$, exemplar nodes C_{ex}
Output: seed sets to expand as focused clusters

$\boldsymbol{\beta} \leftarrow$ InferAttributeWeights(C_{ex})
// (re)-weigh edges by feature similarity of end-nodes
for each $(i, j) \in E$ **do**
  $w(i, j) = 1/(1 + \sqrt{(\boldsymbol{f_i} - \boldsymbol{f_j})^T diag(\boldsymbol{\beta})(\boldsymbol{f_i} - \boldsymbol{f_j})})$
end for
$w' \leftarrow \max_{w'} w' \notin$ distribution$f(\{w|\ w \geq w'\})$
Build induced subgraph $g(V', E', F)$ s.t.
$\forall u, v \in V', (u, v) \in E,\ w(u, v) \geq w'$ iff $(u, v) \in E'$
return ConnectedComponents$(g(V', E', F))$

3.2.3 Focused Outlier Detection

Our algorithm also identifies outlier nodes in each focused cluster along with finding the clusters in a unified fashion. Our definition of a focused cluster outlier is quite intuitive: a node that belongs to a focused cluster structurally (having many edges to its members), but that deviates from its members in some focus attributes significantly is an outlier. To quantify this definition, the main idea is to identify the best *structural* nodes BSNs (best in terms of *un*weighted conductance) during the course of expansion (P3 Lines 3, 14, 22) and later check if there exist any BSNs which were not included in the resulting focused cluster (A1 Line 11).

In order to identify the best structural node for a cluster in each iteration, we need to also track its unweighted conductance. An advantage of our proposed approach is that the overhead of computing the unweighted $\Delta\phi$ of a node, in addition to its weighted $\Delta\phi^{(w)}$, is negligible. The reason

[2]P4 Line 5 removes a node even if conductance remains the same, however, the number of such steps is also bounded by cluster size.

Procedure P3 EXPAND

Input: attributed graph $G(V, E, F)$, focused cluster C, set BSN, current conductance $\phi_{curr}^{(w)}$

Output: a focused cluster C, its best structural nodes BSN, and its conductance $\phi_{curr}^{(w)}$

1: **repeat**
2: $bestNode = NULL$,
3: $bestStructureNode = NULL$
4: $candidateNodes \leftarrow neighbors(C)$
5: $\Delta\phi_{best}^{(w)} = 0$, $\Delta\phi_{best} = 0$
6: **for each** node n in $candidateNodes$ **do**
7: $\Delta\phi_n^{(w)}, \Delta\phi_n \leftarrow$ GETΔCONDUCTANCE$(G, C, n,$ADD$)$
8: **if** $\Delta\phi_n^{(w)} < \Delta\phi_{best}^{(w)}$ **then**
9: $\Delta\phi_{best}^{(w)} = \Delta\phi_n^{(w)}$
10: $bestNode \leftarrow n$
11: **end if**
12: **if** $\Delta\phi_n < \Delta\phi_{best}$ **then**
13: $\Delta\phi_{best} = \Delta\phi_n$
14: $bestStructureNode \leftarrow n$
15: **end if**
16: **end for**
17: **if** $bestNode \neq NULL$ **then**
18: $C \leftarrow C \cup bestNode$
19: $\phi_{curr}^{(w)} = \phi_{curr}^{(w)} + \Delta\phi_{best}^{(w)}$
20: **end if**
21: **if** $bestStructureNode \neq NULL$ **then**
22: $BSN \leftarrow BSN \cup bestStructureNode$
23: **end if**
24: **until** $bestNode = NULL$
25: **return** $C, BSN, \phi_{curr}^{(w)}$

is that, to compute $\Delta\phi$, we simply count the total number, instead of the total weight, of those same edges that are involved in the computation of $\Delta\phi^{(w)}$. As such, both conductances can be computed efficiently at the same time and the best structural node and the best focused cluster node can be identified simultaneously.

3.2.4 Complexity analysis

Given an attributed graph $G(V, E, F)$ and the exemplar nodes C_{ex}, we first create similar and dissimilar node pairs which we use to infer the attribute weights. As the optimization objective we adopt is convex and as we aim for a diagonal solution, local-optima-free gradient descent techniques will take $O(\frac{d}{\epsilon^2})$ for an ϵ-approximate answer [6].

To determine good core sets to expand clusters around, we re-weigh the graph edges by the weight vector $\boldsymbol{\beta}$ with complexity $O(dm)$. Assuming $\boldsymbol{\beta}$ is sparse with only a few non-zero entries for focus attributes, the multiplicative factor becomes effectively constant yielding a complexity of $O(m)$. Next, we identify the top-k edges with largest weights on which we induce G to find the core sets ($k \ll m$). To do so, we use a min-heap to maintain this top set while making a single pass over the edges. This requires $O(m \log k)$ in the worst case, assuming each edge triggers an insertion into the heap. Using these top-k edges we estimate the parameters of a Normal distribution, which takes $O(k)$. Next we make another pass over the edge set and subject each to a membership test against the Normal model in $O(m)$. We induce the graph on all the edges that pass the test, the connected components of which yield the core sets. Overall complexity for finding the core sets is thus $O(m \log k)$.

Procedure P4 CONTRACT

Input: attributed graph $G(V, E, F)$, focused cluster C, current conductance $\phi_{curr}^{(w)}$

Output: a focused cluster C and its conductance $\phi_{curr}^{(w)}$

1: **repeat**
2: $removed \leftarrow false$
3: **for each** node n in C **do**
4: $\Delta\phi_n^{(w)} \leftarrow$ GETΔCONDUCTANCE$(G, C, n,$REMOVE$)$
5: **if** $\Delta\phi_n^{(w)} \leq 0$ **then**
6: $C \leftarrow C \backslash n$
7: $\phi_{curr}^{(w)} = \phi_{curr}^{(w)} + \Delta\phi_n^{(w)}$
8: $removed \leftarrow true$
9: **end if**
10: **end for**
11: **until** $removed = false$
12: **return** $C, \phi_{curr}^{(w)}$

For expanding a focused cluster, we enlist all the non-member neighbors as the candidate set C and evaluate their weighted Δ conductance. As discussed in §3.2.2, the complexity is $\sum_{n \in C} d(n)$. Since $C \subseteq V$, it is equivalently $O(m)$. As we add one node at each iteration, the total complexity becomes $O(|S|m)$ where $|S|$ is the size of the focused cluster, and $|S| \ll n$. Also note that focused clusters can be extracted around each core set in parallel.

We remark that scanning candidate set C for picking the best node takes $O(m)$ in the worst case. Assuming small rounded focused clusters, one can expect that not all edges of G are "touched" by C's neighbors. This implies sub-linear performance for expanding a single cluster in practice.[3]

3.2.5 Variants of FOCUSCO

We conclude this section by briefly discussing a couple of variants of our problem setting and how we can adapt our proposed algorithm to handle these variants.

In one variant of the problem, the user might explicitly ask for the exemplar nodes s/he provided to be included in the focused clusters. While it is highly likely that most of the exemplar nodes will indeed be part of the focused clusters found in Algorithm 1, inclusion of all is not guaranteed. To handle this setting, we can include the connected components induced on the exemplar nodes as additional core sets (in P2 Line 7) and later never allow the removal of any exemplar node in extracting the clusters (in P4 Lines 5-9).

Another variant involves the user asking for a sparser representation of the focus attributes. In other words, it may be practical to define the similarity among the exemplar nodes using as few attributes as possible, especially in high dimensions. In such a case, we can tune the regularization constant γ that we introduced in Equation (1) for learning the weight vector $\boldsymbol{\beta}$. Specifically, a large γ drives the second term in the objective to become large. To make the pairs in P_D as dissimilar as possible, a dense $\boldsymbol{\beta}$ is learned. The smaller the γ gets, the less the emphasis on the dissimilar pairs becomes, and a sparser weight vector is learned.[4]

In other settings, the user may choose to explicitly provide the set of dissimilar nodes or the attribute relevances (i.e., the $\boldsymbol{\beta}$ vector) directly, which can be incorporated trivially.

[3]Different ways of choosing the core sets (e.g., only using user-provided nodes) can remove the dependence on processing the entire graph, and allow FOCUSCO to run in sublinear time.

[4]For $\gamma = 0$, constraint on P_D is completely waived and $\boldsymbol{\beta}$ is zero.

4. EVALUATION

In this section we thoroughly evaluate our method[5] on clustering quality, outlier detection performance, and runtime on synthetic and real-world networks. None of the existing methods address the focused clustering and outlier detection problem we pose in this paper. Nevertheless, we compare to two representative techniques, CODA [14] and METIS [19]. CODA is a graph clustering and outlier detection algorithm on attributed graphs, and treats all attributes equally. The clustering is not steered by user-preference, as such, it clusters the whole graph. METIS is a graph partitioning algorithm and does not provide outlier detection. Both methods expect the number of clusters as input.

To evaluate *focused clustering* quality, we use the Normalized Mutual Information (NMI), a widely used metric for computing clustering accuracy of a method against the desired ground truth [21]. The ground truth in our case is the true focused clusters that are relevant to a particular user's interest. The best NMI score is 1. We evaluate *outlier detection* performance by the F1-score; the harmonic mean of precision and recall for a known set of outliers.

4.1 Results on Synthetic Graphs

4.1.1 Data generation

To study the behavior of our algorithm compared to other approaches on graphs with ground truth (focused) clusters and outliers, we generated synthetic graphs with various number of clusters focusing on different subsets of the attribute space, containing various number of clusters, and with varying size ranges. Our generative algorithm is based on the planted partitions model [9].

Simply put, given the desired number of nodes in each cluster we split the adjacency matrix into blocks defined by the partitioning. For each block B_{ij}, we choose a probability p_{ij}. Using a random draw process we assign a 1, i.e. an edge, for each possible entry in the block, and 0 otherwise. In other words, p_{ij} specifies the density of each block. The diagonal blocks constitute the actual clusters and off-diagonal entries yield the cross edges. Unless otherwise noted, we set $p_{ii} = 0.35$ and $0.10 \leq p_{ij} \leq 0.25$, $i \neq j$.

We assign the graph clusters generated, either to one of two focus attribute sets (i.e. focus-1 or focus-2) or as unfocused. Please note that in real-world graphs, we expect to see more than two focuses on a variety of attribute subspaces. For each focused cluster, one of the two subsets (focus-1 or focus-2) is chosen as focus attributes. For each attribute i in this subset the attribute values are drawn from a Normal distribution $N(\mu_i, \sigma)$ with uniform random mean $\mu_i \in [0, 1]$ and a variance $\sigma = 0.001$. The variance is specifically chosen to be small such that the clustered nodes "agree" on their focus attributes. The rest of the attributes, on the other hand, are drawn from a Normal distribution with much larger variance; $N(0, 1)$. In contrast, all of the attribute values of nodes in unfocused clusters are drawn from large-variance Normals.

Focused outliers are generated by randomly choosing members from each focused cluster and "deflating" (depending on the setting) one or more of their focus attributes i; by replacing them by a value drawn from $N(\mu_i, \sigma = 1)$.

[5]A FocusCO implementation is available at `http://bit.ly/focusedclustering`

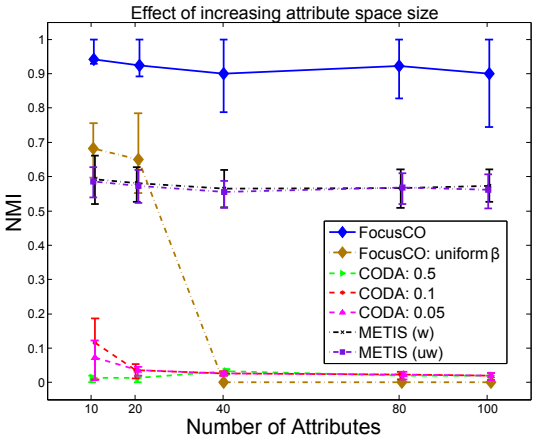


Figure 2: NMI vs. attribute size $|F|$. Results averaged over 100 runs, bars depict 25-75%.

4.1.2 Clustering quality

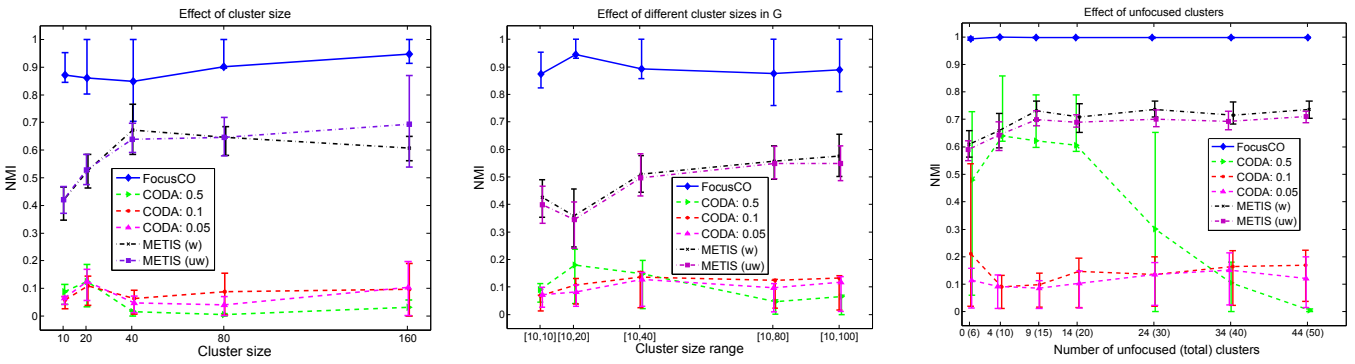
To study the clustering performance, we generated graphs with 3 focused clusters that are coherent in focus-1 attributes, 3 focused clusters that are coherent in focus-2 attributes, and 3 unfocused clusters, for a total of 9 clusters. The task is to extract the focus-1 clusters with the respective user preference.

For comparison we use CODA and METIS, which do not perform focused cluster extraction. They both partition the entire graph, and thus, we explicitly need to select the 3 best clusters that these methods returned, by measuring the overlap among the clusters they produced to the ground-truth clusters. Since both methods require the number of clusters to be provided as input, we asked for the correct number of (9) clusters, i.e. best performance, although in practice this number is hard to choose as the number of hidden clusters is unknown, especially for large graphs.

METIS is a graph partitioning algorithm that does not handle attributes. In one version of METIS, we ignore the attributes and use only the graph structure. In a second version, we incorporate attribute information by weighing the edges of the graph (using $\boldsymbol{\beta}$) by the attribute similarity of their end nodes. We call these two versions as weighted METIS (w) and unweighted METIS (uw). While CODA can perform clustering for attributed graphs, a main challenge with it is to carefully choose a λ parameter that controls the trade-off between structural and attribute similarity of the nodes. In our experiments we report results using 3 different λ settings, 0.05, 0.1, and 0.5, for CODA.

Figure 2 shows the clustering performance (mean NMI over 100 independent runs) of the methods when we increase the number of attributes while retaining the same number of (5) focus attributes for the focused clusters. We observe that FocusCO remains superior to all the other approaches in the face of irrelevant attributes for the clustering task.

To illustrate the importance of weight learning, we also study the performance of a variant of our FocusCO, in which we use a uniform attribute weight vector $\boldsymbol{\beta}$, i.e., we bypass weight learning and directly perform cluster extraction. We observe that the performance of this version of our algorithm drops quickly with increasing attribute size. Our analysis suggests that this occurs due to the edge weights having a more and more uniform distribution when weighted by using a uniform $\boldsymbol{\beta}$, which yields an inferior collection of



(a) NMI vs. cluster size (b) NMI vs. variable cluster size (c) NMI vs. number of unfocused clusters

Figure 3: NMI clustering quality results on synthetic graphs, for FocusCO, CODA with three different λ parameter settings, and METIS on un/weighted graphs; (a) for changing cluster sizes (all clusters have the same size), (b) for changing cluster size variance (graph has variable size clusters), and (c) for increasing number of unfocused clusters. FocusCO performs the best in all scenarios across a wide range of settings. Symbols depict the mean over 100 runs, bars depict 25-75%.

core sets around which we find clusters. Thus, we proceed with studying the performance of our original FocusCO.

Next in Figure 3 we show the clustering performance of the methods under various other settings. In (a), we increase the cluster size where we create clusters of the same size in the graph. In (b), we allow the graph to contain variable size clusters and increase the variance of the cluster sizes, by randomly drawing them from increasing ranges. Finally in (c), we increase the number of unfocused clusters in the graph, while keeping the number of focused clusters fixed.[6] Notice that recovering a few focused clusters of interest in the existence of more unfocused clusters is an increasingly challenging problem.

From all these setups, we observe that FocusCO outperforms the competing methods and their variants in all scenarios. Weighted METIS seems to achieve slightly better performance than the unweighted version, although the differences are not significant. CODA's accuracy is the lowest, as the homophily assumption it is making does not hold in the full attribute space. We note that in (c), one parameterization of CODA ($\lambda = 0.5$) achieves as high accuracy as METIS for small number of clusters. Its accuracy, however, quickly drops when many more unfocused clusters than focused ones are introduced. Other two parameterizations give low accuracy, pointing out the sensitivity of CODA to the choice of its parameter.

Finally, we study the clustering performance when focus-1 and focus-2 clusters share common focus attributes. We create 20 node attributes out of which the first 10 are assigned as focus-1 attributes and the next 10 are assigned as focus-2 attributes to the respective clusters. Then, we gradually overlap the focus attributes until they are the same set of 10 for all the focused clusters. Figure 4 shows that the performance of FocusCO remains stable and high across all overlaps. The accuracy of METIS (w) is also quite stable and stays around 0.6 (METIS (uw) is not expected to be affected in this setup). We also notice that CODA's performance starts increasing after more than 50% of the focus attributes overlap. At 100%, there is essentially only a single focus in the graph, where CODA's performance peaks. This suggests that CODA is more suitable for attributed graphs with uniform graph clusters and would suffer when the graph contains many heterogeneous focused clusters (with multiple focuses) as we would expect to see in real networks.

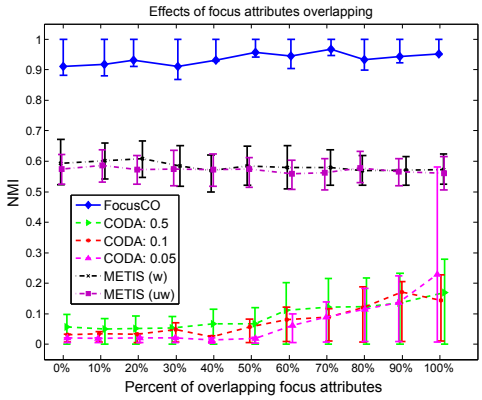


Figure 4: Clustering performance by increasing overlap on focus attributes of different focused clusters. (mean NMI over 100 runs, bars: 25-75%).

These results show the robustness of FocusCO, where its performance remains quite stable across different settings. They also illustrate that the general graph clustering methods are not suitable for our focused clustering problem, as their performance is (*i*) sensitive to the (parameter) setting, and (*ii*) lower than that of FocusCO at all settings.

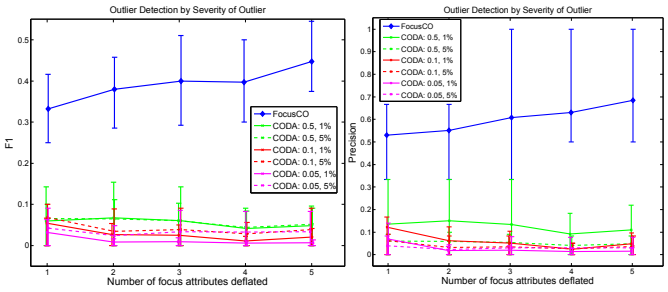
4.1.3 *Outlier detection*

Next we evaluate outlier detection performance. Since METIS does not detect outliers, we compare to CODA. In addition to its λ parameter, CODA expects a parameter r that controls the top percentage of nodes to be returned as outliers. We report experiments for the cross-product of $\lambda = \{0.05, 0.1, 0.5\}$ and $r = \{1\%, 5\%\}$.

In the first setup, we study the performance with respect to the severity of outliers. If an outlier deviates in a larger number of focus attributes from its cluster members, it becomes more severe in outlierness, but easier to detect. To create outlier nodes with higher outlierness, we gradually in-

[6] To ensure sparsity of the growing graphs, we set $p_{ij} = 0.02$.

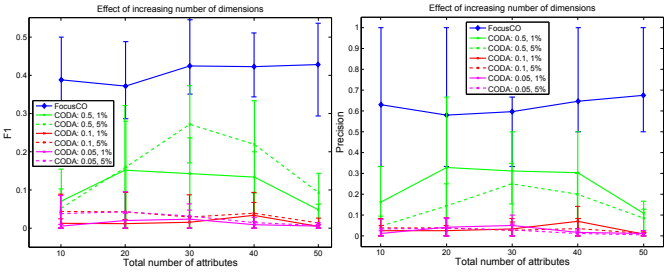
crease their number of focus attributes that we deflate. Figure 5 shows the F1-score and precision (averaged over 100 runs). We observe that FocusCO achieves superior performance to CODA in both metrics. We can also notice the increase in detection accuracy of FocusCO with increasing number of deflated focus attributes (i.e., increasing ease in spotting outliers), as we expected, while the same trend is not apparent for CODA potentially because it does not calibrate to attribute subspaces.



(a) F1-score (b) precision

Figure 5: Outlier detection performance by increasing number of deflated focus attributes.

We further analyze the outlier detection accuracy with respect to the attribute space size. In Figure 6 we observe that CODA's performance starts dropping after a certain number of attributes, where identifying descriptive focus attributes becomes more and more crucial to accurately spot the outliers that are hidden in different community structures. The performance of FocusCO on the other hand remains quite stable and superior to CODA.



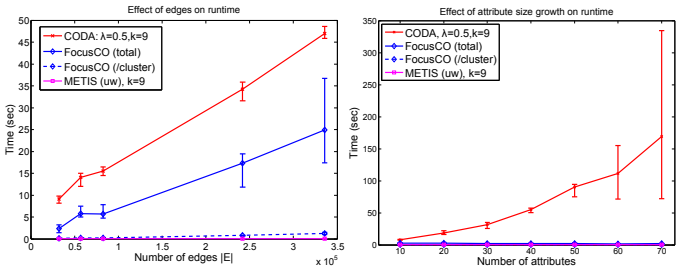
(a) F1-score (b) precision

Figure 6: Outlier detection performance by increasing number of attributes $|F|$.

Finally, we note that the precision of FocusCO is often higher than its recall, while both being superior to CODA's.

4.1.4 Scalability

Finally we study the scalability of the methods. Figure 7 shows the running times with increasing number of edges and attributes. CODA's inference techniques are computationally demanding, and thus, its running time is much higher than other methods. METIS on the other hand uses an extremely efficient heuristic and achieves low running times. We report the average cluster extraction time for FocusCO in addition to total running time, as each cluster extraction can be performed in parallel. In fact, we notice that while its total running time increases by graph size, average time to extract a single cluster remains stable and low. In such a case, FocusCO is also comparable to METIS.



(a) time vs. #edges (b) time vs. #attributes

Figure 7: Running time scalability w.r.t. (a) number of edges $|E|$ and (b) number of attributes $|F|$.

4.2 Results on Real-world Graphs

4.2.1 Dataset description

We use several attributed networks obtained from real-world data to evaluate our approach. Disney is an *Amazon* co-purchase graph of Disney movies.[7] Each movie has 28 attributes such as price, rating, number of reviews, etc. PolBlogs is the citation network among a collection of online blogs that discuss political issues. Attributes are the keywords in their text. DBLP and 4Area are two different co-authorship networks of computer science authors. The attributes reflect the conferences which an author has published in broadly (DBLP), or just in databases, data mining, information retrieval, and machine learning (4Area). Finally, we have the friendship relations of YouTube users and the attributes depict their group memberships. Dataset statistics are given in Table 2.

Table 2: Real-world datasets used in this work. Average running time in seconds per cluster ± std (avg. number of clusters extracted).

Dataset	$\|V\|$	$\|E\|$	$\|F\|$	Running time (sec)
Disney	124	333	28	0.0017 ± 0.0022 (8.3)
PolBlogs	362	1288	44839	0.0040 ± 0.0052 (2.8)
4Area	27199	66832	4	0.0052 ± 0.0018 (3390.9)
DBLP	30599	146647	18	0.2868 ± 0.0630 (761.4)
YouTube	77381	367151	30087	2.9643 ± 0.7201 (257.4)

4.2.2 Running time

Our real datasets come from various domains and have different node, edge, and attribute counts. Here we report running time experiments to demonstrate the efficiency of our method on these real graphs.

We setup 10 runs of our method on each graph, each with a randomly sampled 1% of the attributes as the focus attributes. Each run returns a different number of clusters, thus we report the running time per cluster averaged over the 10 runs and their standard deviations in Table 2. Notice that similar to Figure 7, the running times are quite low. In particular, the average time to extract a cluster in our largest graph YouTube takes around 3 seconds.

4.2.3 Case Studies

The first case study we consider is finding two types of focused clusters in Disney. In one instance, a user wants to understand how the popularity of a movie influences its

[7] http://www.ipd.kit.edu/~muellere/consub/

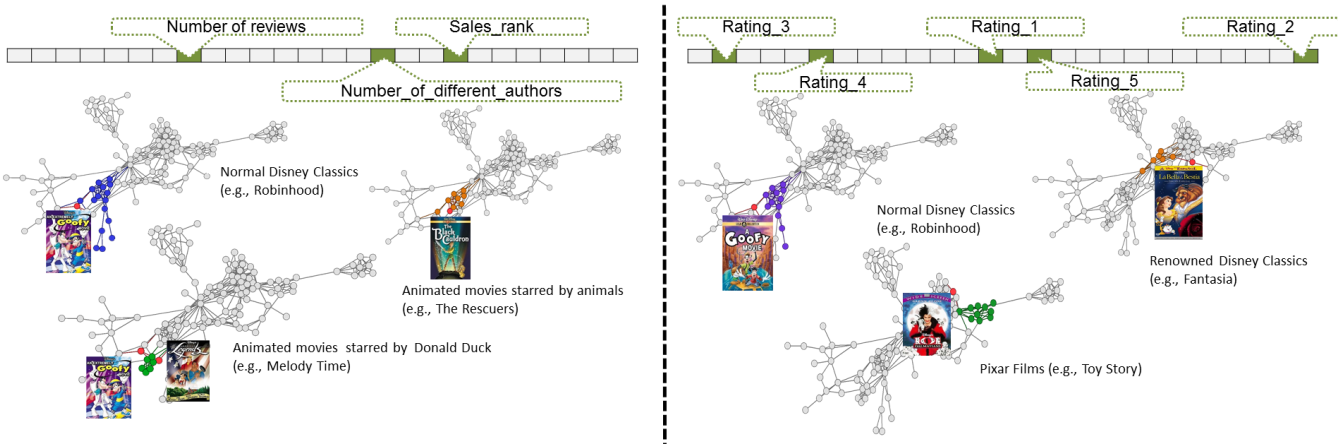


Figure 8: Two sets of focused clusters in Disney. Left clusters focus on attributes related to popularity (sales rank, number of reviews, etc.) Nodes in right clusters share similar ratings. Outliers are marked with red and illustrated. See text for discussion. (best viewed in color)

community in a co-purchase network. The user decides that the product's popularity is related to the features `Number_of_reviews` and `Sales_rank`, and so chooses a few products which have similar values in those attributes. FocusCO then uses this exemplar set C_{ex} to learn an attribute weighting $\boldsymbol{\beta_u}$. This $\boldsymbol{\beta_u}$ reflects the user's intent, and has also captured another dimension correlated with those attributes; `Number_of_different_authors`.

Several extracted focused clusters for this task are shown on the left in Figure 8. In general, the discovered clusters consist of movies of similar age and acclaim. The first focused cluster (blue) reflects traditional Disney classics such as `Robinhood`. Its outlier is a sequel (`An Extremely Goofy Movie`) that is much less popular than the other classics in the cluster. The second community (green) focuses on popular older Disney movies, and has outliers such as `American Legends` and again the Goofy sequel, that are much less popular. The third cluster (orange) *overlaps* with the first focused cluster. It is a subset of the classic Disney movies of the larger cluster that were predominantly starred by animals (e.g., `The Rescuers`). Its cluster outlier is `The Black Caldron`, which although of similar vintage, starred a human protagonist and was much less popular.

In the next instance, a user wants to examine how the differences in the distribution of consumer ratings affect the Disney clustering. In this case, the focused clusters represent collections of movies that are similarly rated (e.g., Pixar films or animated Disney classics). The outliers represent movies which are rated differently from the movies they are purchased with, and reflect consumer opinion. The results are shown in the right of Figure 8. The first focused cluster (purple) represents traditional Disney classics. Its outlier (which was included in the previous focused popularity cluster) is the movie `A Goofy Movie` which although reasonably popular, is not as high rated. The second cluster (green) is the Pixar community, featuring high-rated movies like `Toy Story`. Its focused outlier is the live action version of `101 Dalmatians`, a movie which is rated quite differently than most Pixar films. The third cluster consists of renowned Disney films such as `Fantasia`. It also contains an outlier, the Spanish version of `Beauty and the Beast`.

We consider a second case study on PolBlogs, where a user seeks to understand the difference between blog content

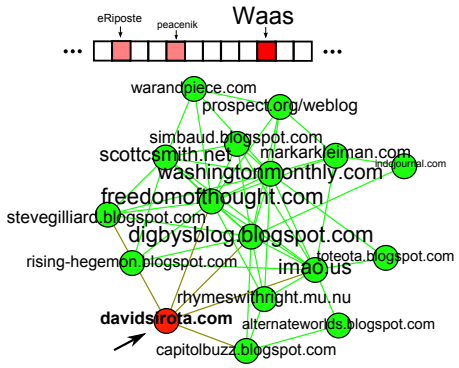


Figure 9: A focused cluster of liberal blogs in PolBlogs with a focus on Iraq war debate. Outlier `David Sirota` does not mention `Waas` in his posts.

written by different liberal bloggers during the height of the Iraq war controversy in 2005. Using several liberal blogs as an example set, the user learns a $\boldsymbol{\beta_u}$ which represents the focus of the bloggers (as shown in the top of Figure 9, text size of features proportional to weight). It contains words such as `eriposte`, an active liberal blogger, `peacenik`, a term for an anti-war activist, and most strongly `Waas`. Murray Waas was an independent journalist praised for his investigative journalism of the Bush administration.[8] The focused community outlier for this group is David Sirota, a well-connected liberal blogger who did not explicitly mention Waas in the dataset.

The third case study we consider is on the 4Area dataset. Here a user wants to understand who the outliers are in the data mining co-authorship network. S/he learns a $\boldsymbol{\beta_u}$ using `{Christos Faloutsos, Jiawei Han, Jon M. Kleinberg, Jure Leskovec, Andrew Tomkins}` as input, who focus primarily on data mining. One of the focused clusters found is around data mining researchers mostly in industry, as shown in Figure 10. The outlier is `Cameron Marlow`, the former head of Facebook's data science team, who collaborated with the researchers in the data mining community but published on information retrieval.

[8] `http://www.huffingtonpost.com/jay-rosen/murray-waas-is-the-woodwa_b_18875.html`

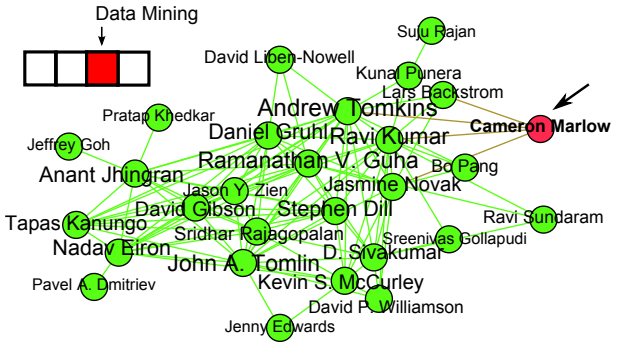


Figure 10: A focused cluster of `data mining` researchers in 4Area. Outlier `Cameron Marlow` closely publishes with them, but on `information retrieval`.

5. CONCLUSION

In this work we introduce a new problem of finding *focused clusters and outliers* in large attributed graphs.

Given a set of exemplar nodes that capture the user interest, our goal is two-fold: 1) "chop out" clusters of similar nodes that are densely connected and exhibit coherence in a subset of their attributes, called the focus attributes, and 2) identify focused outliers, i.e. nodes that belong to a focused cluster in network structure but show deviance in some focus attribute(s). We propose an efficient algorithm that infers the focus attributes of interest to the user, and that both extracts focused clusters and spots outliers simultaneously. Experiments on synthetic and real-world graphs show the effectiveness and scalability of our approach and that the existing graph clustering and outlier detection techniques are not suitable to handle the newly posed problem.

Acknowledgments

The authors thank the anonymous reviewers for their helpful comments. This material is based upon work supported by the ARO Young Investigator Program grant with Contract No. W911NF-14-1-0029, an ONR SBIR grant under Contract No. N00014-14-P-1155, NSF Grant IIS-1017181, the Stony Brook University Office of Vice President for Research, the Young Investigator Group program of KIT as part of the German Excellence Initiative, and by a postdoctoral fellowship by the Flanders research foundation (FWO). Any findings and conclusions expressed in this material are those of the author(s) and do not necessarily reflect the views of the funding parties.

6. REFERENCES

[1] L. Akoglu, M. McGlohon, and C. Faloutsos. OddBall: Spotting anomalies in weighted graphs. In *PAKDD*, 2010.

[2] L. Akoglu, H. Tong, B. Meeder, and C. Faloutsos. PICS: Parameter-free identification of cohesive subgroups in large attributed graphs. In *SIAM SDM*, pages 439–450, 2012.

[3] R. Andersen, F. Chung, and K. Lang. Local graph partitioning using pagerank vectors. In *FOCS*, 2006.

[4] A. Banerjee, S. Basu, and S. Merugu. Multi-way clustering on relation graphs. In *SIAM SDM*, 2007.

[5] S. Basu, I. Davidson, and K. Wagstaff. *Clustering with Constraints: Algorithms, Applications and Theory*. 2008.

[6] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, NY, USA, 2004.

[7] D. Chakrabarti, S. Papadimitriou, D. S. Modha, and C. Faloutsos. Fully automatic cross-associations. In *KDD*, pages 79–88, 2004.

[8] A. Clauset. Finding local community structure in networks. *Physical Review E*, 72(2):026132, 2005.

[9] A. Condon and R. M. Karp. Algorithms for graph partitioning on the planted partition model. *Random Struct. Algorithms*, 18(2):116–140, 2001.

[10] I. Dhillon, S. Mallela, and D. Modha. Information-theoretic co-clustering. In *KDD*, 2003.

[11] C. H. Q. Ding, X. He, H. Zha, M. Gu, and H. D. Simon. A min-max cut algorithm for graph partitioning and data clustering. In *ICDM*, 2001.

[12] G. W. Flake, S. Lawrence, and C. L. Giles. Efficient identification of web communities. In *KDD*. 2000.

[13] J. Gao, H. Cheng, and P.-N. Tan. Semi-supervised outlier detection. In *ACM Symp. on Appl. Comp.*, 2006.

[14] 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.

[15] D. F. Gleich and C. Seshadhri. Vertex neighborhoods, low conductance cuts, and good seeds for local community methods. In *KDD*, pages 597–605, 2012.

[16] S. Günnemann, I. Färber, B. Boden, and T. Seidl. Subspace clustering meets dense subgraph mining: A synthesis of two paradigms. In *ICDM*, 2010.

[17] D. Hanisch, A. Zien, R. Zimmer, and T. Lengauer. Co-clustering of biological networks and gene expression data. In *ISMB*, pages 145–154, 2002.

[18] P. Iglesias, E. Müller, F. Laforet, F. Keller, and K. Böhm. Statistical selection of congruent subspaces for outlier detection on attributed graphs. In *ICDM*, 2013.

[19] G. Karypis and V. Kumar. Multilevel algorithms for multi-constraint graph partitioning. In *Proc. of Supercomputing*, pages 1–13, 1998.

[20] B. Long, Z. Zhang, X. Wu, and P. S. Yu. Spectral clustering for multi-type relational data. In *ICML*, 2006.

[21] C. D. Manning, P. Raghavan, and H. Schtze. *Intro to Information Retrieval*. Cambridge University Press, 2008.

[22] F. Moser, R. Colak, A. Rafiey, and M. Ester. Mining cohesive patterns from graphs with feature vectors. In *SDM*, 2009.

[23] E. Müller, P. I. Sanchez, Y. Mülle, and K. Böhm. Ranking outlier nodes in subspaces of attributed graphs. In *ICDE Workshops*, pages 216–222, 2013.

[24] A. Y. Ng, M. I. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In *NIPS*, 2001.

[25] C. C. Noble and D. J. Cook. Graph-based anomaly detection. In *KDD*, pages 631–636, 2003.

[26] J. Tang and H. Liu. Unsupervised feature selection for linked social media data. In *KDD*, pages 904–912, 2012.

[27] Y. Tian, R. A. Hankins, and J. M. Patel. Efficient aggregation for graph summarization. In *SIGMOD*, 2008.

[28] H. Tong and C.-Y. Lin. Non-negative residual matrix factorization with application to graph anomaly detection. In *SIAM SDM*, pages 143–153, 2011.

[29] F. Wang and J. Sun. Distance metric learning in data mining. In *SIAM SDM*. Tutorials, 2012.

[30] J. Whang, D. Gleich, and I. Dhillon. Overlapping community detection using seed set expansion. In *CIKM*, pages 2099–2108, 2013.

[31] S. White and P. Smyth. A spectral clustering approach to finding communities in graph. In *SDM*, 2005.

[32] E. P. Xing, A. Y. Ng, M. I. Jordan, and S. J. Russell. Distance metric learning with application to clustering with side-information. In *NIPS*, pages 505–512, 2002.

[33] J. Yang and J. Leskovec. Overlapping community detection at scale: a nonnegative matrix factorization approach. In *WSDM*, pages 587–596, 2013.

[34] Y. Zhou, H. Cheng, and J. X. Yu. Graph clustering based on structural/attribute similarities. *PVLDB*, 2(1):718–729, 2009.