Detecting Anomalous Subgraphs on Attributed Graphs Using Graph Cuts

Mahito Sugiyama[1] Keisuke Otaki[2]

[1] ISIR, Osaka University, 8-1, Mihogaoka, Ibaraki-shi, Osaka, 567-0047, Japan
[2] Graduate School of Informatics, Kyoto University, Yoshida-Honmachi, Sakyo-ku, Kyoto, 606-8501, Japan

Abstract. Detecting anomalies from structured *graph* data is becoming a critical task for many applications such as an analysis of disease infection in communities. To date, however, there exists no efficient method that works on massive *attributed graphs* with millions of vertices for detecting anomalous subgraphs with an abnormal distribution of vertex attributes. Here we report that this task is efficiently solved using the recent graph cut-based formulation, which has been originally used for feature selection on networks. We thoroughly examine the method using various sizes of synthetic and real-world datasets and show that our method is more than five orders of magnitude faster than the state-of-the-art method and is more effective.

1 Introduction

Anomaly detection is one of crucial tasks in data mining as anomalous objects (*outliers*) causes serious problems across applications [1]. Despite recent development of anomaly detection methods on multivariate datasets [5, 6, 23, 25] that efficiently find sparsely populated objects, anomaly detection on structured data, in particular on *graphs*, is still developing. The main difficulty is accounting for *inter-dependencies* between objects to find anomalous regions in which objects are connected to each other, which makes the task of anomaly detection on graphs largely different from that on multivariate data of multi-dimensional feature vectors.

Plain graphs with no attribute information on vertices have been firstly investigated [2, 7], where the objective of anomaly detection is to find anomalous regions that have rare structural patterns. Now rapid technological advances produce massive amount of *attributed graphs*, where each vertex is associated with a label/attribute, and the objective becomes to detect densely connected subgraphs in which a distribution of attributes is significantly different from the rest of the region. For example in an analysis of disease infection on a social network, each person is annotated as whether or not he/she is already infected and, to understand the cause of disease infection, our goal is to detect an anomalous local community with the high rate of infected people.

To date, several methods including the state-of-the-art gAnomaly [17] have been proposed that try to solve the above task of detecting anomalous subgraphs on attributed graphs. However, the following problems remain unsolved: (1) Scalability; massive graphs with millions of vertices cannot be treated in a reasonable time; (2) Cardinality constraint; the size of anomalous subgraphs cannot be specified by the user, which is an important requirement in real-world applications.

We overcome in this paper the above problems using the recently proposed graph cut-based formulation [4]. The method, called SConES, has been originally used for feature selection on networks. The key strategy is a minimum-cut reformulation, which enables us to solve the problem by the *maximum flow algorithms* in an efficient and exact manner. Although the original SConES cannot directly handle the size constraint on the resulting subgraph, we solve the problem by applying the *parametric flow algorithm* by Gallo et al. [10] in optimization, which gives the entire *regularization path* along with changes in a regularization parameter. This means that we can pick up the best solution that fulfills the size constraint from the set of possible solutions.

This paper is organized as follows: We introduce our method in Section 2, discuss related work in Section 3, evaluate it by experiments in Section 4, and conclude the paper in Section 5.

2 Anomalous Subgraph Detection

Given an attributed weighted graph $G = (V, E, A)$, where V is a set of vertices, E a set of edges, and $A : V \to \mathbb{A}$ a function that maps each vertex to a value of either binary ($\mathbb{A} = \{0, 1\}$) or real-valued ($\mathbb{A} = \mathbb{R}$). This function defines the *degree of anomalousness* of each vertex, that is, a vertex v is anomalous if $A(v) = 1$ in the binary case and is more anomalous for large $A(v)$ in the real-valued case.

Our ultimate goal is to find a subset $S \subset V$ which maximizes the sum of values $\sum_{v \in S} A(v)$ under two constraints that the elements of S are *connected to each other* and the cardinality $|S|$ is the user specified number k. Unfortunately, this problem is infeasible in practice since the maximum-weight connected graph (MCG) problem:

$$\operatorname*{argmax}_{S \subset V} \sum_{v \in S} A(v) \text{ such that } G[S] \text{ is connected and } |S| = k$$

is known to be strongly NP-complete [16] ($G[S]$ denotes the subgraph of G induced by $S \subset V$). In the following we solve the problem efficiently by focusing on local connectivity rather than conducting an exhaustive search over all connected subgraphs.

2.1 Formulation

To achieve our objective and find anomalous subgraphs, here we define the following problem based on the SConES formulation [4]: Given an attributed graph $G = (V, E, A)$, the *anomalous subgraphs finding problem* is to find the optimal subgraph $G[S]$ induced by a subset $S \subset V$, which is the solution of

$$\max_{S \subset V} \sum_{v \in S} A(v) - \lambda \sum_{e \in C(S)} w(e) - \eta\,|S|, \tag{1}$$

where $w(e)$ is the weight of the edge e,

$$C(S) = \{\, \{v, u\} \in E \mid v \in V \setminus S,\, u \in S \,\}$$

is the set of edges that have one endpoint in S (i.e., the *cut set* of S), and λ and η are two real-valued parameters. The first term is for quantifying the anomalousness of a subgraph $G[S]$, which coincides with the cardinality of the set $\{\, v \in S \mid A(S) = 1 \,\}$ in the binary case. The second and third terms are penalties, where the second is to enforce the *connectivity* of S as it penalizes selecting a vertex without selecting all of its neighbors, and the third is to enforce the *sparsity*.

The notable advantage of this formulation is that this is exactly and efficiently solved by the *maximum flow algorithm*. For a graph $G = (V, E, A)$, we construct an s-t graph $G' = (V', E')$ as follows: $V' = V \cup \{s, t\}$ by adding a source node s and a sink node t, $E' = E \cup \{\, \{u, v\} \mid u \in \{s, t\}, v \in V \,\}$, and the capacity $c : E' \to \mathbb{R}$ is given as

$$c(\{u, v\}) = \begin{cases} A(v) - \eta & \text{if } u = s \text{ and } v \in V, \\ \eta - A(v) & \text{if } u = t \text{ and } v \in V, \\ \lambda\, w(\{v, u\}) & \text{if } u, v \in V. \end{cases}$$

Note that edges whose capacities are negative are ignored.

Theorem 1 ([4]). *Given an attributed graph G. Let $(S \cup \{s\}, V \setminus S \cup \{t\})$ be the minimum s-t cut of the s-t graph G'. The set S is the solution of the problem* (1) *on G.*

Since the s-t mincut problem is solved by a *maximum flow algorithm*, the optimal subgraph $G[S]$ is exactly obtained by simply applying a maximum flow algorithm to the transformed s-t graph G', whose time complexity is $O(nm \log(n^2/m))$ [12], where $n = |V'|$ and $m = |E'|$.

2.2 Size Constraint on Anomalous Subgraphs

In the formulation (1), which is fundamentally the same as SConES, it is not intuitive how to set two parameters, in particular η that controls the size of S. However in anomalous subgraph detection, it is desirable to allow the user to input the constraint on the size of subgraphs. Here we achieve this requirement by solving the following modified problem:

$$\begin{aligned} &\max_{S \subset V,\ \eta \in \mathbb{R}} \sum_{v \in S} A(v) - \lambda \sum_{e \in C(S)} w(e) - \eta\,|S|, \\ &\text{subject to: } |S| \leq k. \end{aligned} \tag{2}$$

Interestingly, we can obtain all possible minimum cuts simultaneously along with changes of η without increasing the time complexity $O(nm \log(n^2/m))$. This is achieved by applying the *parametric flow*

algorithm presented by Gallo et al. [10][3] since the s-t graph G' always becomes a *parametric network* with respect to the regularization parameter η. This is a specific type of networks satisfying the following three conditions: (1) the capacity $c(\{s, v\}) = A(v) - \eta$ is non-decreasing as $-\eta$ increases for all $v \in V$; (2) the capacity $c(\{t, v\}) = \eta - A(v)$ is non-increasing as $-\eta$ increases for all $v \in V$; and (3) the capacity $c(\{u, v\}) = \lambda\, w(\{u, v\})$ is constant for all $u, v \in V$. It is known that in such a case the maximum flow value takes a continuous piecewise linear function of η, and hence the optimal cut (subgraph) does not change between any pair of breakpoints.

The parametric flow algorithm produces the sequence of all optimal solutions (subsets of vertices) $S_1, S_2, \ldots, S_l$. These solutions always have the *nesting property*:

$$\emptyset \subset S_1 \subset S_2 \subset \cdots \subset S_{l-1} \subset S_l \subset V$$

with decreasing the corresponding parameter values $\eta_1, \eta_2, \ldots, \eta_l$. The optimal solution of the problem (2) is therefore computed by simply choosing S_i such that $|S_i| \leq k$ and $|S_{i+1}| > k$.

Furthermore, this hierarchical structure can be used for *visualization* of anomalous regions, based on the intuition that a smaller subgraph with a larger regularization parameter η is more anomalous than a larger subgraph. In particular, we design a scoring function for vertices as follows: Let us denote by $i(v)$ a natural number such that $v \notin S_{i(v)-1}$ and $v \in S_{i(v)}$ for any vertex $v \in V$. Define a scoring function $q : V \to [0, 1]$ as

$$q(v) := \frac{l - i(v) + 1}{l}.$$

This visualization reveals the *hierarchical structure* of anomalous regions.

3 Related Work

Anomaly detection on graphs is roughly divided into two settings: on plain (unlabeled) graphs and attributed (labeled) graphs (a detailed survey is given by [3]).

On plain graphs, the objective is to detect regions that have rare structural patterns. Various approaches have been proposed [2, 7, 13, 14, 18], for example, Akoglu et al. [2] introduced the concept of an *egonet*, which is a subgraph with its neighbors, and measured the abnormality of vertices by checking whether their egonets obey some power-low extracted from real-world graph data.

Followed by studies on plain graphs, anomaly detection on attributed graphs have been also heavily studied. Noble and Cook [21] were the first to investigate anomaly detection on attributed graphs, where the Minimum Description Length (MDL) principle was used to define abnormal substructures through measuring the compression quality of frequent subgraphs. Eberle and Holder [9] also tried to define the degree of anomalousness based on the structure of subgraphs with their attributes. Gao et al [11] introduced *community outliers*, which significantly deviate from the rest of the local community members, and proposed an algorithm CODA to find them, but the algorithm strongly depends on the initialization step and the convergence is not guaranteed [3]. A node outlier ranking technique GOutRank was proposed by Müller et al. [19], although it does not aim at finding densely connected subgraphs. The concept of *focused* clustering and outlier detection was introduced by Perozzi et al. [22], where only clusters and outliers focused by the user through their exemplars are detected. Recently, Li et al. [17] tries to find anomalous subgraphs by estimating probability distributions of anomalous attributes by the EM algorithm. The method, called gAnomaly, is the state-of-the-art and will be compared to our proposed method in the next section.

Note that clustering techniques on attributed graphs (e.g. GBAGC [27]), which do not focus on detecting anomalies, can be used for anomaly detection as the task of anomaly detection can be achieved by dividing the whole graph into two clusters of normal and abnormal sets.

4 Experiments

We examine our method, which we call PFAG, on synthetic and real-world graph data compared to the state-of-the-art methods gAnomaly [17] and GBAGC [27].

[3] This fact is pointed out in [24] but has not been used in any applications. A similar result is theoretically analyzed in [15].

4.1 Experimental Methods

Environment. We used Mac OS X version 10.9.4 with 3.5 GHz Intel Core i7 CPU and 32 GB of memory. We implemented PFAG and gAnomaly in R, version 3.1.1, where PFAG calls the parametric flow algorithm[4] written in C++ and compiled by gcc version 4.9.0. Note that the most expensive optimization part of gAnomaly is implemented in C, hence comparison of running time between PFAG and gAnomaly is fair. For GBAGC, the official implementation[5] was used.

Datasets. We generated various sizes of attributed graph datasets in the following manner: Fist, we generated graphs according to the Watts-Strogatz network model [26] using the R package `igraph`. Second, we took the largest dense subgraph using a method proposed by Clauset et al. [8] and assumed that this community is an anomalous subgraph, that is, a vertex is labeled as 1 if it is in the subgraph and 0 otherwise. Although our method can handle real-valued attributes, we examine only the binary case as gAnomaly cannot treat real-valued attributes.

We used three real-world datasets: CORA[6], DBLP, and Amazon. DBLP and Amazon were obtained from SNAP[7]. In CORA, we used the largest cluster "Neural Networks" as an anomalous subgraph, which is the same protocol as in [17]. In DBLP and Amazon, we chose the largest community given by [28] and assigned it as anomalous. Moreover, we used a small subset of DBLP (denoted as DBLP(s)) by taking the four largest communities. Their sizes are summarized in Table 1.

Evaluation. To investigate the performance of detection methods, in each synthetic and real-world graph dataset, we randomly chose 20 % of vertices from the anomalous subgraph and assigned the label 0 to them. The task is to recover these hidden anomalous vertices. Precision and recall were computed, and the F-measure was also computed from them to summarize the performance. In addition, the modularity [20] was computed to evaluate the goodness of division of resulting subgraphs, and the gain of the modularity was obtained. We report the mean and the standard deviation of these values in 20 repeats.

4.2 Results and Discussion

Efficiency. First we examine the efficiency using synthetic datasets. Figure 1 shows results of running time of each method with respect to the number of vertices. The number of edges are fixed as twice the number of vertices. We could not use GBAGC when the number of vertices is larger than 10^5 as it run out of memory.

We can clearly see that our method PFAG is much faster than gAnomaly, and it is the only method that can be applied to large graphs with more than 10^5 vertices in a reasonable time. In real datasets in Table 1, our method is also the fastest and more than five orders of magnitude faster than gAnomaly.

Sensitivity. Next we analyze the sensitivity of our method with respect to changes in the parameter λ. Since gAnomaly also has a regularization parameter λ, we also analyze the sensitivity of gAnomaly and compare it to our method. We used synthetic graphs of 1000 vertices and 2000 edges. We applied GBAGC to synthetic data, but it could not find any anomalous subgraph in any setting, that is, all vertices are always in the same cluster. The reason might be that their framework is too sensitive to labels of neighbors and cannot handle our setting.

Results are plotted in Figure 2. These plots show that our method is robust to changes in the parameter λ if it is smaller than 1, and is not stable if it is large. This is why our method chooses too small anomalous subgraphs if λ is large, and both precision and recall become lower while the modularity increases. If λ is sufficiently small (around 0.1), the performance of PFAG in terms of both the F-measure and the modularity is always better than gAnomaly. These results indicate that we do not need to be carefully tune the parameter λ and just set to a small value. In the following, we always set $\lambda = 0.01$ in PFAG.

[4] Source code: `http://research.microsoft.com/en-us/downloads/d3adb5f7-49ea-4170-abde-ea0206b25de2/`. Since the code can handle only integers for parameters, we first transform every parameter to an integer by multiplying some constant value.

[5] `http://www.cais.ntu.edu.sg/~chi/software.html`

[6] `http://www.cs.umd.edu/~sen/lbc-proj/LBC.html`

[7] `http://snap.stanford.edu/index.html`

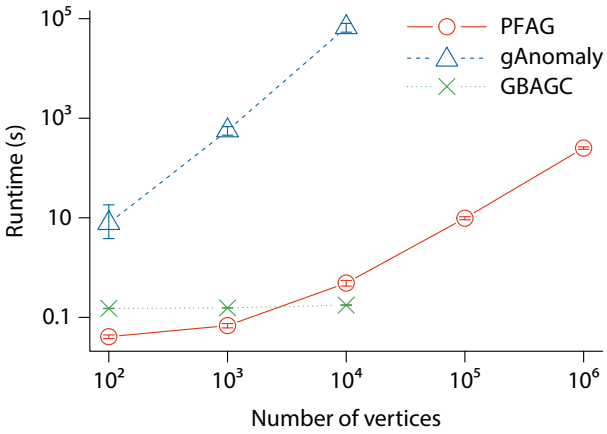

Fig. 1. Running time (seconds).

Table 1. Results on real-world datasets.

Data	n	m	F-measure			Gain of modularity			Runtime (s)		
			PFAG	gAnomaly	GBAGC	PFAG	gAnomaly	GBAGC	PFAG	gAnomaly	GBAGC
CORA	2708	5429	**0.915**	0.892	0.112	0.0624	**0.107**	−0.272	**0.124**	32861	0.358
DBLP(s)	3194	8714	**0.904**	0.775	0.129	0.0591	**0.0846**	−0.0310	**0.171**	39450	0.279
Amazon	334863	925872	**0.951**	—	—	**0.0782**	—	—	**26.6**	—	—
DBLP	317080	1049866	**0.848**	—	—	**0.113**	—	—	**48.6**	—	—

In gAnomaly, if the parameter λ is small (from 0.01 to 1), it cannot regularize the detection, that is, it just chooses vertices labeled as anomalous. But once λ becomes larger than 1, regularization effect is suddenly too strong and the performance gets worse in terms of both the F-measure and the modularity. When λ is larger than 30, it did not find any anomalous subgraph. This result indicates that in practice it is not easy to find a good setting of λ. In the following, we always set $\lambda = 1$ in gAnomaly.

Effectiveness. Finally, we investigate the performance on various sizes of synthetic graphs and real-world graphs. Results on synthetic data are shown in Figure 3 and those on real data are in Table 1. Moreover, we show visualization of the CORA dataset by our method in Figure 4. We can see that our method PFAG is always superior to gAnomaly in terms of both the F-measure and the modularity on every data size. In real data, our method shows the best scores on all datasets in the F-measure, while gAnomaly is the best in the modularity. However, note that gAnomaly is not scalable and takes more than five orders of magnitude slower than PFAG. The clustering method GBAGC shows the worst score on every dataset.

5 Conclusion

In this paper we have presented a scalable method PFAG, which detects anomalous subgraphs from attributed graphs. This method is based on the SConES formulation [4], thereby it is exactly and efficiently solved by the maximum flow algorithms through a mincut reformulation. Moreover, using the parametric flow algorithm [10], we have achieved to introduce the cardinality constraint, that is, the user can specify the desirable number of vertices. Experiments have shown that our method is much faster than the state-of-the-art method gAnomaly and is more effective on synthetic and real graph datasets.

Currently, PFAG can handle only one-dimensional attributes, while some methods including GBAGC can use *multi-dimensional* attributes. Thus extending our formulation to multi-dimensional attributes, that is, how to design the attribute function A, is an interesting future work.

6 Acknowledgment

This work was partially supported by JSPS KAKENHI 26880013 and Grand-in-Aid for JSPS Fellows 26-4555.

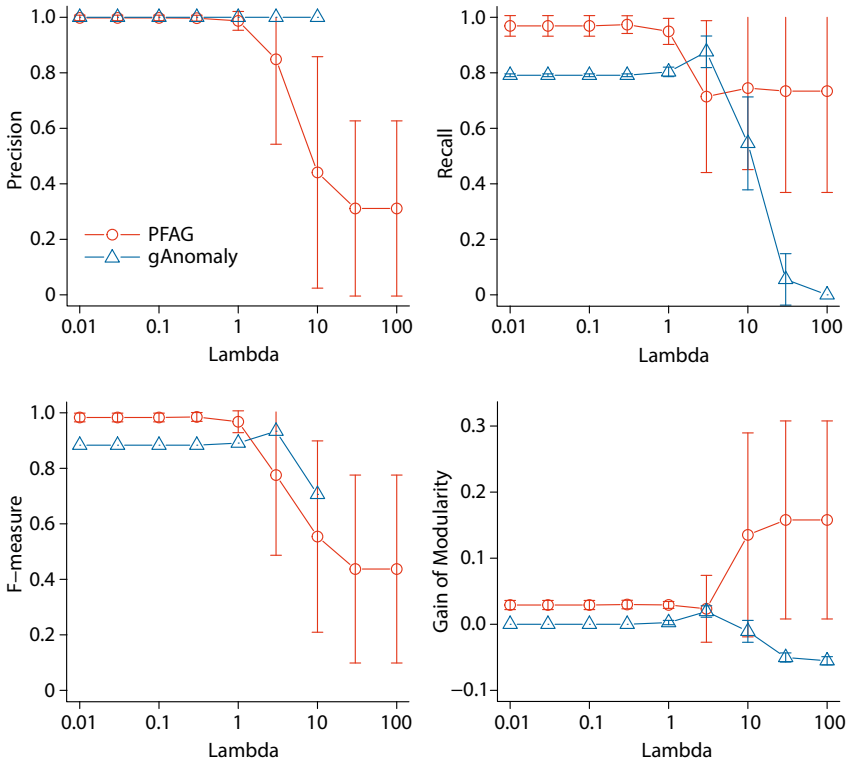

Fig. 2. Performance with respect to changes in regularization parameter λ.

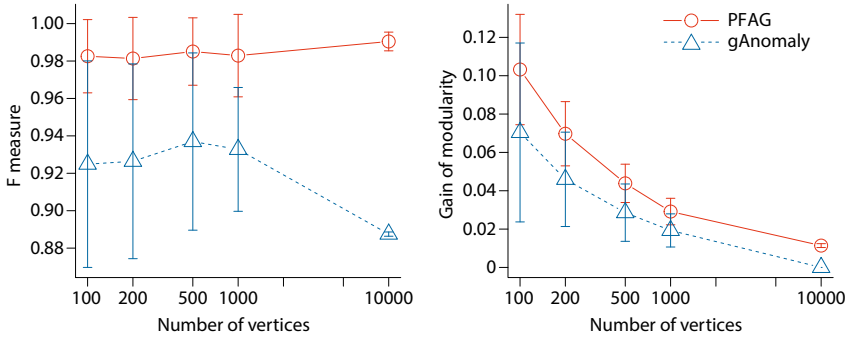

Fig. 3. Performance on synthetic data.

References

1. Aggarwal, C.C.: Outlier Analysis. Springer (2013)
2. Akoglu, L., McGlohon, M., Faloutsos, C.: OddBall: Spotting anomalies in weighted graphs. In: PAKDD. LNCS, vol. 6119, pp. 410–421. Springer (2010)
3. Akoglu, L., Tong, H., Koutra, D.: Graph based anomaly detection and description: a survey. Data Mining and Knowledge Discovery pp. 1–63 (2014)
4. Azencott, C.A., Grimm, D., Sugiyama, M., Kawahara, Y., Borgwardt, K.M.: Efficient network-guided multi-locus association mapping with graph cuts. Bioinformatics 29(13), i171–i179 (2013)
5. Bhaduri, K., Matthews, B.L., Giannella, C.R.: Algorithms for speeding up distance-based outlier detection. In: Proceedings of the 17th ACM SIGKDD Conference on Knowledge Discovery and Data Mining. pp. 859–867 (2011)
6. Breunig, M.M., Kriegel, H.P., Ng, R.T., Sander, J.: LOF: Identifying density-based local outliers. In: Proceedings of the ACM SIGMOD International Conference on Management of Data. pp. 93–104 (2000)
7. Chakrabarti, D.: AutoPart: Parameter-free graph partitioning and outlier detection. In: PKDD. LNCS, vol. 3202, pp. 112–124. Springer (2004)
8. Clauset, A., Newman, M.E.J., Moore, C.: Finding community structure in very large networks. Physical Review E 70(6), 066111 (2004)
9. Eberle, W., Holder, L.: Discovering structural anomalies in graph-based data. In: ICDM Workshop. pp. 393–398 (2007)

10. Gallo, G., Grigoriadis, M.D., Tarjan, R.E.: A fast parametric maximum flow algorithm and applications. SIAM Journal on Computing 18(1), 30–55 (1989)
11. Gao, J., Liang, F., Fan, W., Wang, C., Sun, Y., Han, J.: On community outliers and their efficient detection in information networks. In: KDD. pp. 813–822 (2010)
12. Goldberg, A.V., Tarjan, R.E.: A new approach to the maximum-flow problem. JACM 35(4), 921–940 (1988)
13. Henderson, K., Eliassi-Rad, T., Faloutsos, C., Akoglu, L., Li, L., Maruhashi, K., Prakash, B.A., Tong, H.: Metric forensics: A multi-level approach for mining volatile graphs. In: KDD. pp. 163–172 (2010)
14. Henderson, K., Gallagher, B., Li, L., Akoglu, L., Eliassi-Rad, T., Tong, H., Faloutsos, C.: It's who you know: Graph mining using recursive structural features. In: KDD. pp. 663–671 (2011)
15. Kawahara, Y., Nagano, K.: Structured convex optimization under submodular constraints. In: UAI. pp. 459–468 (2013)
16. Lee, H.F., Dooly, D.R.: Algorithms for the constrained maximum-weight connected graph problem. Naval Research Logistics 43(7), 985–1008 (1996)
17. Li, N., Sun, H., Chipman, K., George, J., Yan, X.: A probabilistic approach to uncovering attributed graph anomalies. In: SDM. pp. 82–90 (2014)
18. Lin, C.Y., Tong, H.: Non-negative residual matrix factorization with application to graph anomaly detection. In: SDM. pp. 143–153 (2011)
19. Müller, E., Sanchez, P.I., Mülle, Y., Böhm, K.: Ranking outlier nodes in subspaces of attributed graphs. In: ICDE Workshop. pp. 216–222 (2013)
20. Newman, M.E.J., Girvan, M.: Finding and evaluating community structure in networks. Physical review E 69(2), 026113 (2004)
21. Noble, C.C., Cook, D.J.: Graph-based anomaly detection. In: KDD. pp. 631–636 (2003)
22. Perozzi, B., Akoglu, L. Sánchez, P.I., Müller, E.: Focused clustering and outlier detection in large attributed graphs. In: KDD (2014)
23. Pham, N., Pagh, R.: A near-linear time approximation algorithm for angle-based outlier detection in high-dimensional data. In: Proceedings of the 18th ACM SIGKDD Conference on Knowledge Discovery and Data Mining. pp. 877–885 (2012)
24. Sugiyama, M., Azencott, C.A., Grimm, D., Kawahara, Y., Borgwardt, K.M.: Multi-task feature selection on multiple networks via maximum flows. In: SDM. pp. 199–207 (2014)
25. Sugiyama, M., Borgwardt, K.M.: Rapid distance-based outlier detection via sampling. In: NIPS. pp. 467–475 (2013)
26. Watts, D.J., Strogatz, S.H.: Collective dynamics of 'small-world'networks. Nature 393(6684), 440–442 (1998)
27. Xu, Z., Ke, Y., Wang, Y., Cheng, H., Cheng, J.: GBAGC: A general bayesian framework for attributed graph clustering. ACM Transactions on Knowledge Discovery from Data 9(1), 5:1–5:43 (2014)
28. Yang, J., Leskovec, J.: Defining and evaluating network communities based on ground-truth. In: ICDM. pp. 745–754 (2012)

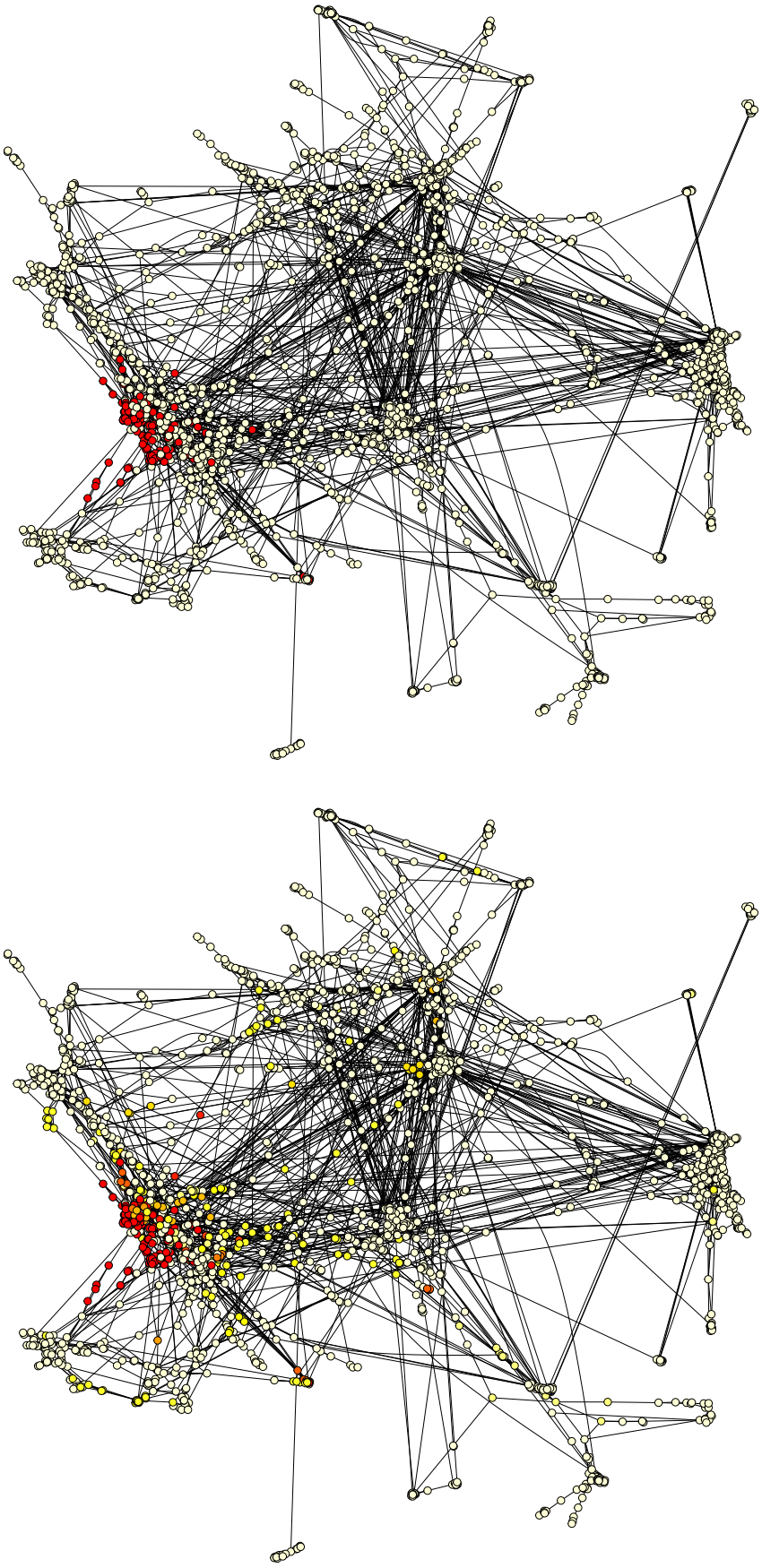

Fig. 4. CORA dataset. (Upper) Anomalous vertices are colored by red. (Lower) Vertices are colored according to the score q obtained by PFAG.