Detecting Anomalous Subgraphs on Attributed Graphs via Parametric Flow

Mahito Sugiyama[1,2] Keisuke Otaki[3]

[1] ISIR, Osaka University, 8-1, Mihogaoka, Ibaraki-shi, Osaka, 567-0047, Japan
[2] JST, PRESTO
mahito@ar.sanken.osaka-u.ac.jp
[3] 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. In particular, the full hierarchy of anomalous subgraphs can be simultaneously obtained via the *parametric flow algorithm*, which allows us to introduce the size constraint on anomalous subgraphs. 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 in detection of anomalous subgraphs.

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, 25, 27] 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 multi-dimensional feature vectors.

Rapid technological advances produce massive amount of *attributed graphs*, where each vertex is associated with a label/attribute, and an anomalous subgraph corresponds to a densely connected region in which a distribution of attributes is significantly different from the rest of the region. Moreover, in many cases, such vertex attribute directly shows whether or not the vertex is anomalous, that is, they can be used as partially supervised information of anomalous subgraphs. Thus, in this situation, our task is to *detect potentially* (*or hidden*) *anomalous regions* on the given graph structure using the attribute information (see Figure 1). This task therefore corresponds to *transductive learning* [8] in the

field of machine learning, where we aim at predicting labels of unlabeled data in a given dataset. For example in an analysis of disease infection on a social network, people are annotated as whether or not they are already infected and, to understand the cause of disease infection and to find potentially infected people, the goal is to detect an anomalous local community with the high rate of infected people.

To date, several methods including the current state-of-the-art gAnomaly [18] have been proposed that try to solve the above task of detecting anomalous subgraphs on attributed graphs. However, the following two 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. In gAnomaly, we have to rerun it many times by changing its parameter in small steps until reaching at a subgraph with the desirable size.

Our goal in this paper is to overcome the above two issues. The key technique of our proposal is to use the recently proposed graph cut-based formulation [4], where the method, called SConES, has been proposed and used for feature selection on networks (weighted graphs). SConES uses the fact that the optimal features correspond to the minimum cut on a given graph, and hence it can be solved by a *maximum flow algorithm* 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* proposed by Gallo et al. [11], which gives the entire *regularization path* along with changes in a regularization parameter. Since the size of subgraphs depends on the regularization parameter, we can pick up the best solution that fulfills the size constraint from the set of possible solutions obtained by the parametric flow algorithm.

This paper is organized as follows: Section 2 describes our method; we first formulate our problem in 2.1 and introduce the cardinality constraint in 2.2, followed by achieving ranking and visualization of anomalous subgraphs in 2.3. Related work is discussed in Section 3, and our proposal is evaluated by experiments in Section 4. We conclude the paper with summarizing our contribution in Section 5.

2 Anomalous Subgraph Detection

Given an weighted graph $G = (V, E)$, where V is a set of vertices and E is a set of edges, and a weight $w(e)$ is assigned to each edge $e \in E$. We consider the situation in which the *degree of anomalousness* of each vertex is given through an *attribute function* A from V, where its range, denoted by $\mathrm{range}(A)$, can be either binary ($\mathrm{range}(A) = \{0, 1\}$) or real-valued ($\mathrm{range}(A) = \mathbb{R}$). In the binary case, a vertex v is *anomalous* if $A(v) = 1$ and is normal if $A(v) = 0$, while v is more and more anomalous if $A(v)$ gets a larger and larger value in the real-valued case. In the following, we treat the graph $G = (V, E)$ and an attribute function A as a triplet $G = (V, E, A)$ and call G an *attributed graph*. In this setting, the

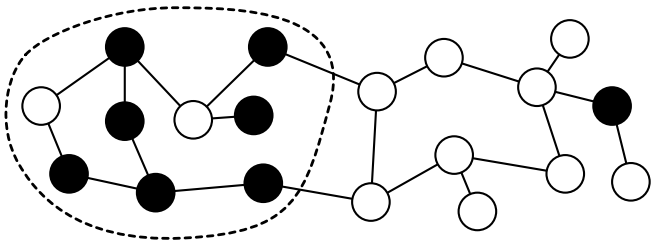

Fig. 1. Our problem setting. Open and filled circles denote normal and anomalous vertices, respectively, given by attributes. Our goal is to find an anomalous subgraph, denoted by a dotted line, where two normal vertices in the subgraph are potential anomalies.

function A can be viewed as partial information of anomalies, and our objective is to recover potentially anomalous regions from together with A and the given network structure G. Notations used in the paper is summarized in Table 1.

One of the most direct mathematical formulation of this problem is as follows: Find a subset $S \subset V$ which maximizes the sum of values $\sum_{v \in S} A(v)$ under two constraints that the vertices in S of G are *connected to each other* by edges and the cardinality $|S| = k$, which is specified by the user. Unfortunately, this problem is infeasible to solve in practice because the maximum-weight connected graph (MCG) problem:

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

is known to be strongly NP-complete [17] ($G[S]$ denotes the subgraph of G induced by $S \subset V$). Thus, instead of tackling this hopeless problem, we focus on *local connectivity* rather than conducting an exhaustive search over all connected subgraphs. Our formulation, which is introduced in the next subsection, allows the user to pick up more than one subgraph at the same time.

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 subgraph 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,

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

Table 1. Notation.

$G = (V, E, A)$	Attributed graph
V	Set of vertices of G
E	Set of edges of G
A	Attribute function from V to $\{0, 1\}$ or $\mathbb{R}$
S	Subset of V
$G[S]$	Subgraph of G induced by S
v, u	Vertex; $v, u \in V$
e	Edge; $e \in E$
$w(e)$	Weight of edge e
$C(S)$	Cut set of S, i.e., $C(S) = \{ \{v, u\} \in E \mid v \in V \setminus S, u \in S \}$
λ	Parameter for connectivity
η	Parameter for sparsity
n	Number of vertices
m	Number of edges
$G' = (V', E')$	s-t graph constructed from G
k	Size constraint (upper bound of the number of vertices of subgraph)
γ	Parameter of parametric network
$S_1, S_2, \ldots, S_l$	Optimal solutions obtained by the parametric flow algorithm
q	Scoring function from V to $[0, 1]$
$i(v)$	Natural number such that $v \notin S_{i(v)-1}$ and $v \in S_{i(v)}$ for all $v \in V$

is the set of edges that have one of two endpoints in S (i.e., the *cut set* of S), and λ and η are two real-valued regularization 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* of the subgraph. Note that SConES has been originally proposed for supervised feature selection on graphs, and we transfer it into the problem of anomalous subgraph detection, where anomalous subgraphs correspond to selected features.

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

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

An example of transformation into an s-t graph is shown in Figure 2. For mathematical convenience, a capacity of an edge can be negative in the above definition. Such edges with negative capacities are ignored in the maximum flow algorithm. We have the following powerful property for this s-t graph.

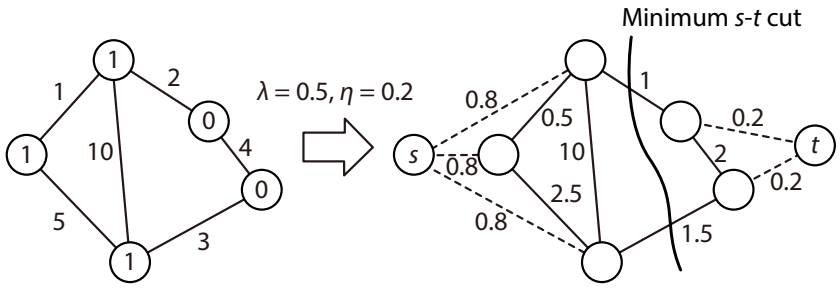

Fig. 2. Example of a graph (left) and its corresponding s-t graph (right) for the maximum flow problem. Numbers in circles denote attribute values $A(v)$ and those on edges denote weights (left) and capacities (right). In this example, $\lambda = 0.5$ and $\eta = 0.2$.

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'. Then the set S coincides with the solution of Problem (1) on G.*

Since the s-t minimum cut problem is solved as a *maximum flow problem*, thanks to the famous max-flow min-cut theorem [23, Chap. 6.1], 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 $O(nm \log(n^2/m))$ [13], where $n = |V'|$ and $m = |E'|$.

2.2 Parametric Flow on Anomalous Subgraphs

In 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:

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

where k is a natural number specified by the user.

Interestingly, we can obtain all possible minimum cuts simultaneously along with changes of the parameter η without increasing the time complexity of the maximum flow algorithm $O(nm \log(n^2/m))$. This is achieved by applying the *parametric flow algorithm* presented by Gallo et al. [11][4] since the s-t graph G' always becomes a parametric network.

A *parametric network* is a specific type of networks equipped with a real-valued parameter γ satisfying the following three conditions:

[4] This fact is pointed out in [26] but has not been used in any applications. A related result is theoretically analyzed in [16].

Algorithm 1 paraAnomaly

Input: Attributed graph $G = (V, E, A)$, size constraint k, connectivity parameter λ
Output: Anomalous subgraph of G
1: Construct the s-t graph G' from G
2: Apply the parametric flow algorithm to G' and obtain the set of optimal solutions $S_1, S_2, \ldots, S_l$ along with changes of the sparsity parameter η
3: Output $G[S_i]$, where $|S_i| \leq k$ and $|S_{i+1}| > k$

1. The capacity $c(\{s, v\})$ is a non-decreasing function of γ for all $v \in V \setminus \{t\}$.
2. The capacity $c(\{t, v\})$ is a non-increasing function of γ for all $v \in V \setminus \{s\}$.
3. The capacity $c(\{u, v\})$ is constant for all $u, v \in V$ with $u \neq s$ and $v \neq t$.

From the definition of the s-t graph G', we can easily confirm the following fact by letting $\gamma = -\eta$.

Lemma 1. *The s-t graph G' is a parametric network with respect to $(-\eta)$.*

Notice that the weight $w(e)$ of edges $e \in E$ is used to construct the s-t graph G', while it is treated as a constant in the parametric network because it is independent from η.

For a parametric network, it is known that the maximum flow value takes a continuous piecewise linear function of $\gamma = -\eta$. Then there must be a finite number of breakpoints $\gamma_1 < \gamma_2 < \cdots < \gamma_{l-1}$, and for each interval $[\gamma_{i-1}, \gamma_i)$, the optimal solution S_i does not change for any $\gamma \in [\gamma_{i-1}, \gamma_i)$. Hence a finite sequence of optimal solutions (subsets of vertices) $S_1, S_2, \ldots, S_l$ is produced by the parametric flow algorithm, where $l-1$ is the number of breakpoints uniquely determined from the property of a given graph.

Here an important property of the sequence of solutions is that they always have the *nesting property*:

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

with increasing the corresponding parameter values $\gamma_1, \gamma_2, \ldots, \gamma_l$ (i.e., decreasing the parameter values $\eta_1, \eta_2, \ldots, \eta_l$ such that $\gamma_i = -\eta_i$). The optimal solution of Problem (2) is therefore computed by simply choosing S_i such that $|S_i| \leq k$ and $|S_{i+1}| > k$. The entire process, which we call *paraAnomaly*, is summarized in Algorithm 1.

2.3 Parametric Flow to Rank

The proposed method paraAnomaly can go one step further: It achieves not only binary discrimination of anomalous subgraphs from the entire graph but also *ranking of anomalous subgraphs*, which is often desirable in anomaly detection. This is directly achieved from the hierarchical structure of the optimal solutions $S_1 \subset S_2 \subset \cdots \subset S_l$, that is, a smaller subgraph with a larger regularization parameter η is more anomalous than a larger subgraph.

Moreover, this ranking can be visualized by designing a scoring function for vertices by focusing on the difference between consecutive subgraphs. 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 vertices $v \in V$. Define a scoring function $q : V \to [0, 1]$ as

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

where the numerator $l - i(v) + 1$ is the number of solutions containing v and the denominator l is the normalizer so that the resulting value $q(v) \in [0, 1]$. Then vertices in a highly anomalous subgraph receives a higher score than those in a low anomalous subgraph. This visualization reveals the hierarchical structure of anomalous regions. We will show an example of visualization of a real-world dataset in Section 4.

3 Related Work

Anomaly detection on graphs is roughly divided into two settings: on plain (unlabeled) graphs and attributed (labeled) graphs. In this section we briefly discuss related work about anomaly detection on graphs, mainly focusing on anomaly detection on attributed graphs, and point out the difference between our method and the existing methods. A comprehensive 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, 14, 15, 19], 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 [22] 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 [10] tried to define the degree of anomalousness based on the structure of subgraphs with their attributes. Gao et al. [12] 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. [20], although it does not aim at finding densely connected subgraphs. The concept of *focused* clustering and outlier detection was introduced by Perozzi et al. [24], where only clusters and outliers focused by the user through their exemplars are detected.

Despite the detailed studies of anomaly detection on attributed graphs, none of the above methods aggressively treat attributes as supervision of anomalousness. This means that their setting is basically *unsupervised*, and hence attributes are not directly associated with the degree of anomalousness. In contrast, recently, Li et al. [18] have considered the transductive setting and tried to recover

anomalous subgraphs from partially labeled vertices by estimating probability distributions of anomalous attributes by the EM algorithm. This is the problem setting that we are considering in this paper, and their method, called gAnomaly, is compared to our proposed method paraAnomaly in the next section as it is the current state-of-the-art.

Clustering techniques on attributed graphs, which do not focus on detecting anomalies, can be used for anomaly detection since the task of anomaly detection can be achieved by dividing the whole vertices into two clusters of normal and abnormal vertices. A representative method GBAGC [29] is also compared to our method in our experiments.

4 Experiments

In this section, we examine our method paraAnomaly on synthetic and real-world graph datasets. First we describe our experimental setting, followed by discussing the results.

4.1 Experimental Methods

Environment. We used Mac OS X version 10.9.4 with a 3.5 GHz Intel Core i7 CPU and 32 GB of memory. Our method paraAnomaly is implemented in R, version 3.1.1, which calls the parametric flow algorithm[5] written in C++ and compiled by gcc version 4.9.0.

Comparison partners. Our main comparison partner is the state-of-the-art method gAnomaly [18]. We re-implemented gAnomaly in R since the official code is not available. Note that the most expensive optimization part of gAnomaly is done by an R function `optim`, in which the core part is implemented in C. Thus comparison of running time between paraAnomaly and gAnomaly is fair. The function $R_N^{(2)}$ (see [18, Equation (3.7)] for its definition) was used as a network regularizer because the author claims that it is more robust to the parameter setting than the other regularize $R_N^{(1)}$.

In addition, a clustering method GBAGC is also included as a comparison partner because it is used as the solo comparison partner of gAnomaly in [18]. The official implementation[6] was used.

Datasets. We generated various sizes of attributed graph datasets in the following manner: First, we generated graphs according to the Watts-Strogatz network

[5] Source code is available at `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.

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

model [28] using the R `igraph` package. Second, we took the largest dense subgraph using a method proposed by Clauset et al. [9] 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 systematically examine only the binary case as gAnomaly cannot treat real-valued attributes.

We used three real-world datasets: CORA[7], DBLP, and Amazon. DBLP and Amazon were obtained from SNAP[8]. In CORA, we used the largest cluster "Neural Networks" as an anomalous subgraph, which is the same protocol as in [18]. In DBLP and Amazon, we chose the largest community given by [30] and assigned it as anomalous. Moreover, we used a small subset of DBLP (denoted as DBLP(s)) by taking the four largest communities. Statistics of datasets are summarized in Table 2.

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. Hence the task is to recover those hidden anomalous vertices from the rest of 80 % anomalous vertices. Precision and recall were computed, and the F-measure was also computed from them to summarize the performance. In addition, we used the gain of the modularity [21] to evaluate the goodness of division of resulting subgraphs without label information. We report the mean and the standard deviation of these values in 20 repeats in every case.

4.2 Results and Discussion

Efficiency. First we examine the efficiency using synthetic datasets. Figure 3 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 finish GBAGC when the number of vertices is larger than 10^5 as it run out of memory. This means that GBAGC cannot treat large graphs, although such graphs are now emerging and needed to be analyzed.

We can clearly see that paraAnomaly 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. The running time scales sub-quadratically with the number of vertices in paraAnomaly, while quadratically in gAnomaly. In real datasets in Table 3, our method is also the fastest and more than five orders of magnitude faster than gAnomaly. We can therefore say that *paraAnomaly is the first method that can handle massive graphs with millions of vertices in detecting anomalous subgraphs.*

Sensitivity. Next we analyze the sensitivity of our method with respect to changes in the parameter λ. Since gAnomaly also has a regularization parameter

[7] `http://www.cs.umd.edu/~sen/lbc-proj/LBC.html`
[8] `http://snap.stanford.edu/index.html`

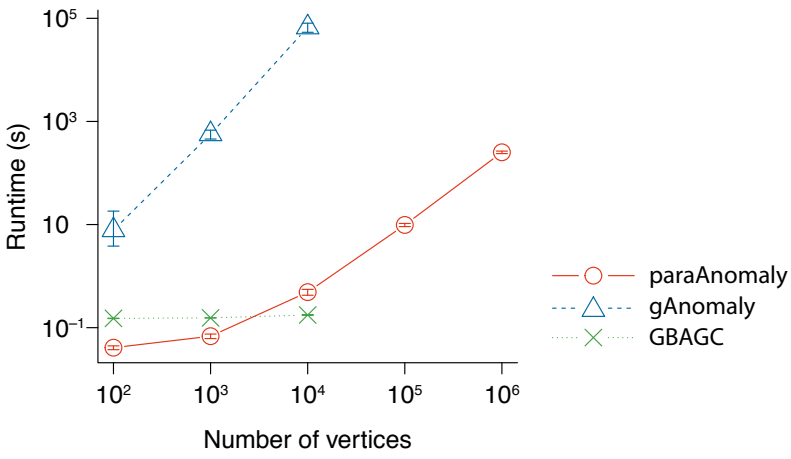

Fig. 3. Running time (seconds). GBAGC run out of memory and broke down when the number of vertices is larger than 10^5.

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 also applied GBAGC to synthetic data, but it could not find any anomalous subgraph in any setting, i.e., 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 4. 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 gets larger than 1. This is why the penalty with respect to the connectivity is too strong, resulting in choosing too small anomalous subgraphs if λ is large. Thus both precision and recall become low values while the modularity increases. If λ is sufficiently small (around 0.1), the performance of paraAnomaly in terms of both the F-measure and the modularity is always better than gAnomaly. In addition, if $\lambda = 0$, precision stays high while recall gets lower again. The reason is that, in such a case, there is no regularization and the method just picks up all given 80 % of vertices and does not pick up any hidden anomalous vertices. To summarize, 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 paraAnomaly.

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 like the case of $\lambda = 0$ in paraAnomaly. But once λ gets larger than 1, regularization effect becomes 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.

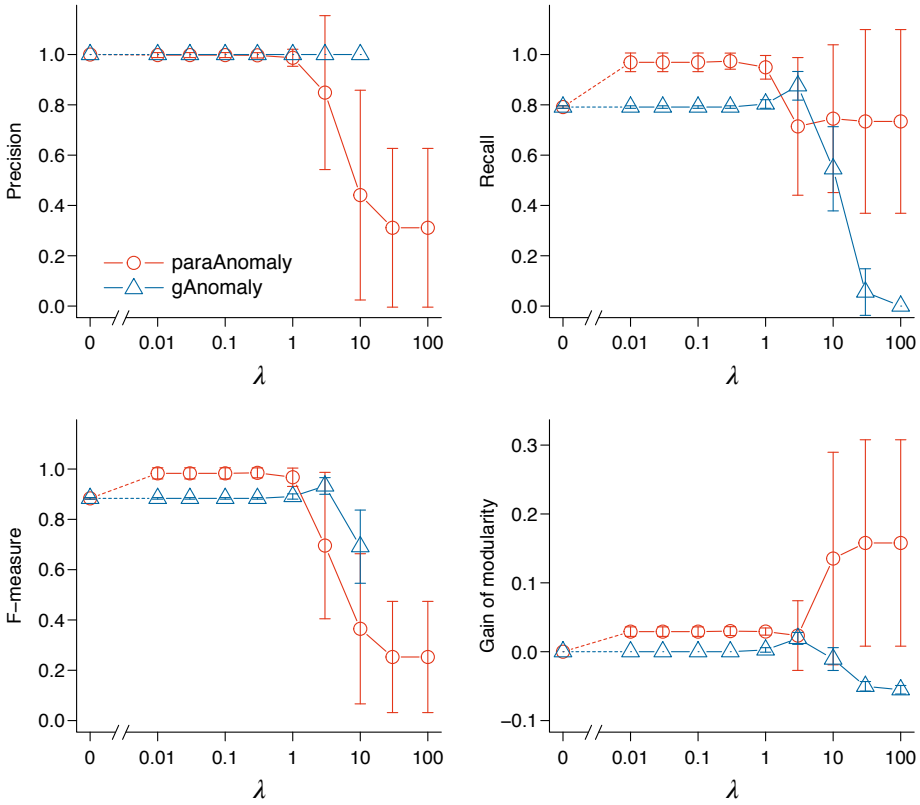

Fig. 4. Performance with respect to changes in regularization parameter λ.

Effectiveness. We investigate the performance on various sizes of synthetic graphs and real-world graphs. Results on synthetic data are shown in Figure 5 and those on real data are in Table 3.

In synthetic data, we can see that our method paraAnomaly is always superior to gAnomaly in terms of both the F-measure and the modularity on every data size. In real data, paraAnomaly 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 paraAnomaly, thereby we could not finish gAnomaly on Amazon and DBLP. The clustering method GBAGC shows the worst score on every dataset. From those results, we can again confirm that our paraAnomaly is the only method that can efficiently and effectively find anomalous subgraphs from large scale graph data.

Visualization. Finally, we demonstrate visualization on the CORA dataset. The original anomalous vertices, corresponds to the cluster "Neural Networks",

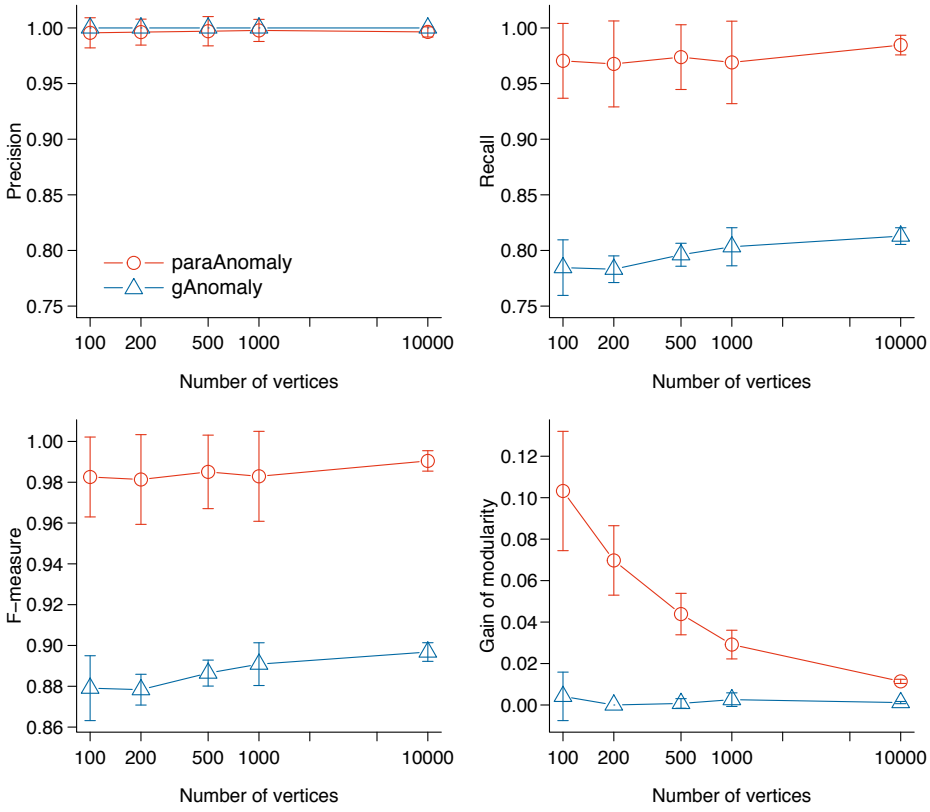

Fig. 5. Performance on synthetic data.

are shown in Figure 6 and the resulting visualization by paraAnomaly is shown in Figure 7. Here we can confirm that vertices that are close to (i.e., densely connected to) anomalous vertices are claimed to be anomalous according to their anomalous scores. Thus, by our method paraAnomaly, one can visualize interesting anomalous communities from the given attributed graphs, which are simultaneously ranked according to their degree of anomalousness.

5 Conclusion

In this paper we have presented a scalable method paraAnomaly, 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 minimum cut reformulation. Moreover, using the parametric flow algorithm [11], we have achieved to introduce the cardinality constraint, that is, the user can specify the desirable number of vertices. Exper-

Table 2. Statistics of real-world datasets. In the table, "Ratio" denotes the ratio of the number of anomalous vertices in each graph.

Data	$\lvert V\rvert$	$\lvert E\rvert$	Average degree	Ratio
CORA	2708	5429	4.010	0.302
DBLP(s)	3194	8714	5.456	0.202
Amazon	334863	925872	5.530	0.160
DBLP	317080	1049866	6.622	0.024

Table 3. Results on real-world datasets. In the table, "paraAno" and "gAno" denote paraAnomaly and gAnomaly, respectively.

Data	**Precision**			**Recall**			**F-measure**		
	paraAno	gAno	GBAGC	paraAno	gAno	GBAGC	paraAno	gAno	GBAGC
CORA	0.969	**0.977**	0.20	**0.867**	0.822	0.078	**0.915**	0.892	0.112
DBLP(s)	**0.955**	0.918	0.16	**0.858**	0.670	0.108	**0.904**	0.775	0.129
Amazon	**0.951**	—	—	**0.951**	—	—	**0.951**	—	—
DBLP	**0.868**	—	—	**0.828**	—	—	**0.848**	—	—

Data	**Gain of modularity**			**Runtime (s)**		
	paraAno	gAno	GBAGC	paraAno	gAno	GBAGC
CORA	0.062	**0.107**	−0.272	**0.124**	32861.436	0.358
DBLP(s)	0.059	**0.085**	−0.031	**0.171**	39450.032	0.279
Amazon	**0.078**	—	—	**26.649**	—	—
DBLP	**0.011**	—	—	**48.626**	—	—

iments 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, paraAnomaly 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

The authors thank Yoshinobu Kawahara for insightful discussions.

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

References

1. Aggarwal, C.C.: Outlier Analysis. Springer (2013)

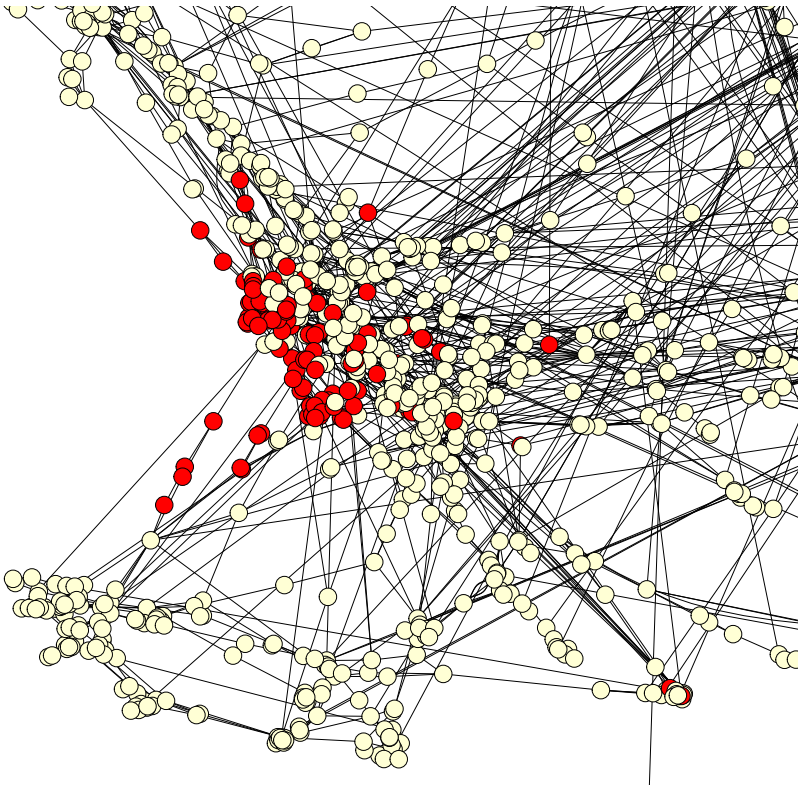

Fig. 6. CORA dataset. Anomalous vertices are colored by red.

2. Akoglu, L., McGlohon, M., Faloutsos, C.: OddBall: Spotting anomalies in weighted graphs. In: Advances in Knowledge Discovery and Data Mining (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: Knowledge Discovery in Databases (PKDD). LNCS, vol. 3202, pp. 112–124. Springer (2004)
8. Chapelle, O., Schölkopf, B., Zien, A.: A discussion of semi-supervised learning and transduction. In: Semi-Supervised Learning, chap. 25, pp. 473–478. MIT Press

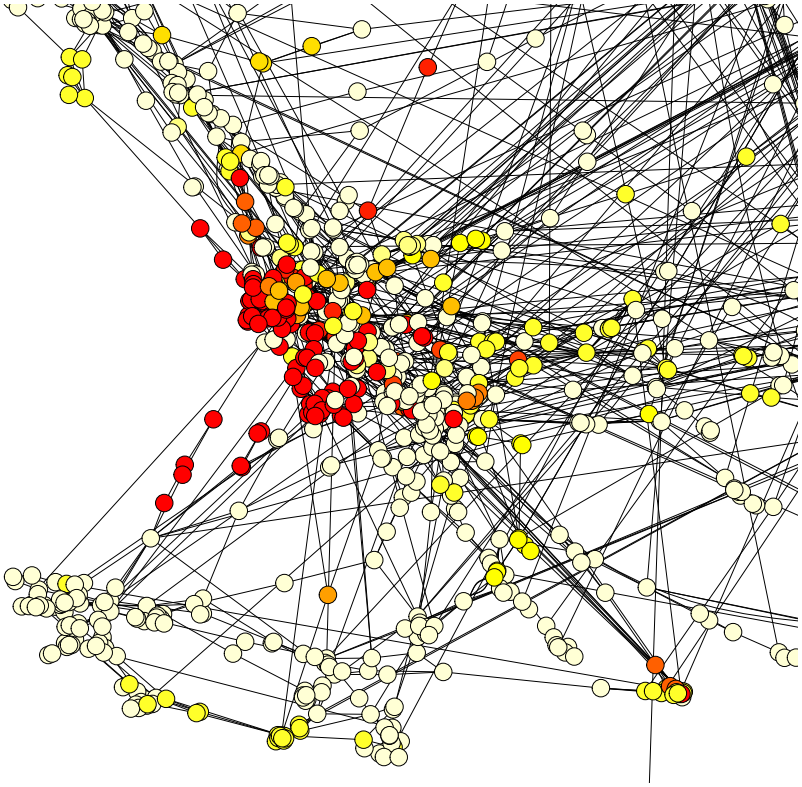

Fig. 7. CORA dataset. Vertices are colored according to the score q obtained by paraAnomaly.

(2006)
9. Clauset, A., Newman, M.E.J., Moore, C.: Finding community structure in very large networks. Physical Review E 70(6), 066111 (2004)
10. Eberle, W., Holder, L.: Discovering structural anomalies in graph-based data. In: IEEE International Conference on Data Mining (ICDM) Workshop. pp. 393–398 (2007)
11. 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)
12. Gao, J., Liang, F., Fan, W., Wang, C., Sun, Y., Han, J.: On community outliers and their efficient detection in information networks. In: Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 813–822 (2010)
13. Goldberg, A.V., Tarjan, R.E.: A new approach to the maximum-flow problem. Journal of the ACM 35(4), 921–940 (1988)
14. 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: Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 163–172 (2010)

15. 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: Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 663–671 (2011)
16. Kawahara, Y., Nagano, K.: Structured convex optimization under submodular constraints. In: Proceedings of Uncertainty in Artificial Intelligence (UAI). pp. 459–468 (2013)
17. Lee, H.F., Dooly, D.R.: Algorithms for the constrained maximum-weight connected graph problem. Naval Research Logistics 43(7), 985–1008 (1996)
18. Li, N., Sun, H., Chipman, K., George, J., Yan, X.: A probabilistic approach to uncovering attributed graph anomalies. In: SDM. pp. 82–90 (2014)
19. Lin, C.Y., Tong, H.: Non-negative residual matrix factorization with application to graph anomaly detection. In: SDM. pp. 143–153 (2011)
20. Müller, E., Sanchez, P.I., Mülle, Y., Böhm, K.: Ranking outlier nodes in subspaces of attributed graphs. In: ICDE Workshop. pp. 216–222 (2013)
21. Newman, M.E.J., Girvan, M.: Finding and evaluating community structure in networks. Physical review E 69(2), 026113 (2004)
22. Noble, C.C., Cook, D.J.: Graph-based anomaly detection. In: Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 631–636 (2003)
23. Papadimitriou, C.H., Steiglitz, K.: Combinatorial Optimization: Algorithms and Complexity. Dover (1998)
24. Perozzi, B., Akoglu, L. Sánchez, P.I., Müller, E.: Focused clustering and outlier detection in large attributed graphs. In: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (2014)
25. 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)
26. Sugiyama, M., Azencott, C.A., Grimm, D., Kawahara, Y., Borgwardt, K.M.: Multi-task feature selection on multiple networks via maximum flows. In: Proceedings of SIAM International Conference on Data Mining (SDM). pp. 199–207 (2014)
27. Sugiyama, M., Borgwardt, K.M.: Rapid distance-based outlier detection via sampling. In: Advances in Neural Information Processing Systems. pp. 467–475 (2013)
28. Watts, D.J., Strogatz, S.H.: Collective dynamics of 'small-world' networks. Nature 393(6684), 440–442 (1998)
29. 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)
30. Yang, J., Leskovec, J.: Defining and evaluating network communities based on ground-truth. In: Proceedings of the 2012 IEEE International Conference on Data Mining (ICDM). pp. 745–754 (2012)