Knowl Inf Syst (2018) 55:571–597
https://doi.org/10.1007/s10115-017-1068-7

REGULAR PAPER

# Graph-based review spammer group detection

**Zhuo Wang**[1] · **Songmin Gu**[1] · **Xiangnan Zhao**[1] · **Xiaowei Xu**[2]

Received: 20 July 2016 / Revised: 22 April 2017 / Accepted: 22 May 2017 /
Published online: 19 July 2017


**Abstract** Online product reviews nowadays are increasingly prevalent in E-commerce websites. People often refer to product reviews to evaluate the quality of a product before purchasing. However, there have been a large number of review spammers who often work collaboratively to promote or demote target products, which severely harm the review system. Much previous work exploits machine learning approaches to detect suspicious reviews/reviewers. In this paper, we introduce a top-down computing framework, namely GGSpam, to detect review spammer groups by exploiting the topological structure of the underlying reviewer graph which reveals the co-review collusiveness. A novel instantiation of GGSpam, namely GSBC, is designed by modeling spammer groups as bi-connected graphs. Given a reviewer graph, GSBC identifies all the bi-connected components whose spamicity scores exceed the given spam threshold. For large unsuspicious bi-connected graphs, the minimum cut algorithm is used to split the graph, and the smaller graphs are further processed recursively. A variety of group spam indicators are designed to measure the spamicity of a spammer group. Experimental study shows that the proposed approach is both effective and efficient and outperforms several state-of-the-art baselines, including graph based and non-graph based, by a large margin.

**Keywords** Fraud detection · Review spam · Review spammer groups · Bi-connected graph · Opinion mining · Graph mining

## 1 Introduction

E-commerce websites often provide rating systems to evaluate the quality of products or services. Such review resources are increasingly influencing consumer's purchase decisions.

✉ Zhuo Wang
zhuowang@sylu.edu.cn

1 Shenyang Ligong University, Shenyang, China

2 University of Arkansas at Little Rock, Little Rock, AR, USA

For the sake of financial profits, some fraudulent reviewers fabricate spam reviews to promote their products or to demote their rivals' products. Such reviewers are called fake reviewers or review spammers, and the products being spammed are called target products [8]. Review spammers often work in groups to fully control the sentiment of the target products, split total effort, and camouflage (i.e., to hide their spamming behaviors by arranging some group members to review irrelevant products or review normally to mislead spam detecting tools) [12,25]. Such review spammer groups are more frequently occurred and are even more harmful than individual review spammers.

Since Jindal and Liu first proposed the fake review/reviewer detection problem in 2008 [8], there have been increasing research interests in this field. Early works focus on individual review spammer detection [6,8,9,11,15]. Recently, there is a trend aiming to detect group spammers [3,5,12,16,19,22,23,25]. In earlier stage, supervised learning-based approaches are adopted, which heavily depend on labeled datasets to train classifiers. However, such methods are shown to be inaccurate due to the fact that there is no ground-truth (labeled) datasets for modeling or evaluation. To bypass the labeling obstacle, many unsupervised models are proposed to detect review spammers, e.g., Markov Random Field (MRF) method [5,6,16].

Compared to individual spammer detection, group spamming detection is not so extensively addressed. Much previous work in group spamming detection exploits frequent itemset mining (FIM) technique to generate candidate spammer groups and then build models to categorize them into spam/non-spam reviewer groups [3,12,22,23]. As stated in [19], such methods can only find *tight* spammer groups in which each group member has to review all the target products. However, group spammers often work in a *looser* manner, i.e., reviewers are not required to review each target product in some group spam campaigns. In our previous work [19], we proposed a graph-based approach to mine loose spammer groups via bipartite graph projection, which can discover high-quality loose spammer groups from only the rating data, i.e., (userid, prodid, rating, timestamp) tuples, no review text is needed. It is well known that review text analysis is computationally inefficient and is often unreliable in review spam detection [14,15,23].

Our previously proposed method in [19], however, suffers from many drawbacks. For instance, the degree of collusion behavior (or similarity) between two reviewers in [19] is simply defined by the number of products that are co-reviewed by the two reviewers in a given time window. Clearly, this relationship metric ignores the review time interval and the rating score deviation, which are two key factors for describing the collusion behavior between two reviewers. Secondly, it generates spammer groups by incrementing the weight threshold to weaken the connectivity of the reviewer graph, so that smaller connected components come into being. In such a method, the granularity used to generate connected components is rough, and the connectivity of a connected component tends to be loose. In this paper, we propose another graph-based approach under a unified top-down computing framework for the review spammer group detection problem. The contribution of this work are three-fold:

– We introduce a top-down computing framework, namely GGSpam (for Graph-based Group SPAM), to detect review spammer groups based merely on the topological structure of the reviewer graph. GGSpam treats the whole review data as a graph structure: reviewers as the nodes, and the collusiveness between reviewers as edges. By adopting a divide-and-conquer strategy, GGSpam recursively breaks the whole graph into small-sized subgraphs. This computing framework has many distinct advantages in comparison with other frameworks in group review spammer detection.

– We propose a novel instantiation of GGSpam, namely GSBC, by modeling review spam groups as bi-connected graphs. GSBC models the collusiveness between two reviewers by considering both the review time interval and the rating score deviation for each common product reviewed by the two reviewers, which can better reflect the colluding behavior between two reviewers. We also design new group spam indicators to evaluate the spamicity of a group from various perspectives.
– We conduct extensive experiments on real-world datasets with/without ground-truth to evaluate the performance of our proposed approach. Experimental results indicate that our proposed method can find high-quality review spammer groups and outperforms several state-of-the-art approaches, including both graph based and non-graph based, by a large margin.

The rest of the paper is organized as follows. Section 2 discusses the related works. Section 3 introduces the GGSpam framework. Section 4 proposes the bi-connected graph-based instantiation of the GGSpam. Section 5 reports the experimental results. We conclude the paper in Sect. 6.

## 2 Related work

Fake review/reviewer detection problem has gained increasing interest in recent ten years. The work can be approximately summarized into three categories: fake review content detection, fake reviewer detection, and fake reviewer group detection. For example, Liu et al. [8] employ (near) duplicate reviews as fake reviews to train classifiers; Ott et al. [15] employ Amazon Mechanical Turks (AMT) to crowd-source fake hotel reviews and use linguistic features analysis of review text to identify deceptive reviews; Lim et al. [11] use behavioral features in rating patterns to spot suspicious reviewers; Wang et al. [18] first introduced review graphs to compute the trustiness of reviewers; Xie et al. [20] detect reviewers who write singleton reviews by temporal analysis.

Recently, there have been increasing works in detecting group review spamming. Mukherjee et al. [12] first introduce FIM technique to generate candidate review spammer groups, which takes reviewer as items, and products as transactions. For example, by setting the minimum support count to 3, they can find candidate spammer groups each contain at least 2 reviewers and each reviewer at least reviews 3 common products. Based on these candidate groups, many computing frameworks are proposed to evaluate the suspicion of each candidate spammer group or individual spammers. For instance, [12] proposed *GSRank* to rank candidate groups using an iterative computing framework which captures the relationship among candidate groups, target products, and individual reviewers. Xu et al. [23] introduce a KNN-based method and a graph-based classification method to predict the spam/non-spam labels for each reviewer belonging to at least one FIM candidate group. Xu et al. [22] propose a statistical model which exploits the EM algorithm to compute the collusiveness of each group member from at least one FIM candidate group.

There are also many other efforts aiming to detect collusive reviewers without using FIM. Leman et al. [1] propose FraudEagle framework, which employs a Loopy Belief Propagation-based (MRF) inference algorithm that solely relies on network effect (the relational structure) among reviewers and products to rank fake reviewers. Spammer groups can be further obtained by doing graph clustering on the induced subgraph which contains the top suspicious reviewers and the corresponding products. Shebuti et al. [16] propose SPEagle, which is extended from FraudEagle by introducing an augmented graph (the review nodes),

and incorporating meta information (e.g., review content, timestamps and star ratings, etc.) as priors, which can greatly improve the ranking precision. Ye et al. [25] propose a two-step method to discover review spammer groups. It first finds out all the suspicious products being spam campaign targets and then clusters spammers on a 2-hop induced subgraph from the top-ranked products. Xu et al. [21] propose FRAUDINFORMER framework to detect colluders via multiple heterogeneous pairwise features extracted from reviewers' rating behaviors and linguistic patterns. A Markov random walk model-based iterative computing framework is exploited to rank reviewers according to their spamicity.

Unlike the above-mentioned group spamming detection frameworks, in [19] we propose a divide-and-conquer-based algorithm (GSBP) which is solely based on the topological structure of the reviewer graph revealing the behavioral similarity between two reviewers. In this paper, we introduce a more general computing framework which is inspired by the basic ideas in [19], and give a full instantiation of the framework. As such, our solution is notably different from, while being complementary to, the previous methods, such as the FIM-based and statistical learning-based approaches.

Review spamming techniques are evolving and vary from different domains. Therefore, there is no *one-size-fits-all* solution in detecting review spamming activities. Although there are a variety of spamming detection techniques proposed by researchers, there is no overall winner in discovering all kinds of spamming strategies. We argue that the best way to improve the detection recall is to incorporate as many techniques as possible.

## 3 The GGS$_{\text{PAM}}$ computing framework

In this section, we give a graph-based unified computing framework, namely GGSPAM, to detect review spammer groups by generalizing the ideas in [19].

GGSPAM computing framework

1. Model a review spammer group $g$ as a sextet form $(R_g, P_g, V_g, S_g, \text{SS}(g), \tau)$, where $R_g$ is the set of review spammers (or members) in group $g$, $P_g$ is the set of target products in group $g$, $V_g$ is the set of reviews written by reviewers in $R_g$ toward products in $P_g$, $S_g$ is the set of spam indicators, $\text{SS}(g)$ is a function of $S_g$ measuring the spamicity of $g$, and $\tau$ is the time window to measure the time closeness degree of co-reviewing
2. Construct a global weighted reviewer graph $G = (R_g, E)$, where $R_g$ is the node set representing reviewers, and $E$ is the edge set whose weights represent the similarity of collusive behaviors for the adjacent reviewers
3. Design a divide-and-conquer algorithm to find all the suspicious review spammer groups whose $\text{SS}(g)$ value is above a given spamicity threshold
4. Rank the detected spammer groups, that is, reorder the detected spammer groups so that more suspicious spammer groups are brought to front and non-spam groups are sent to back to improve the precision at top $k$ position

Our previous work in [19] can be viewed as an instantiation of GGSPAM, which models a spammer group as a $k$-connectivity graph, and defines the similarity between reviewers as the number of products that are co-reviewed by two adjacent reviewers. In this paper, we give another instantiation of GGSPAM based on the bi-connected graph structure, which can better model the reviewer spammer groups and, as a result, significantly improve the quality of the detected spammer groups.

Our graph-based GGSpam framework has many distinct advantages compared to other frameworks. First, GGSpam directly generates holistic spammer groups, comprising group members, the target products, and the spam reviews issued by the group, which provides rich information and context for end-users to analyze and determine the spamicity degree of the detected groups. In comparison, some researchers deal with group spamming problem by only considering detecting individual spammers who are of collusive behaviors [1,23]. For example, [1] exploits MRF to rank the spamicity of reviewers, and for further spotting the spammer groups, an additional stage is required to find review spammer groups by taking into account the top $k\%$ review spammers. Secondly, some supervised or unsupervised methods require that all instances being identically and independently distributed (iid), whereas review spammers usually closely interact with each other in comparison to the genuine reviewers; therefore, machine learning-based approaches which aim to find frequent patterns often are not applicable to this problem [2]. Since review fraud detection problem can be viewed as an anomaly detection problem, our graph-based approach is more suitable for this task because it can capture the underlying co-reviewing relationships among reviewers via the products they review. Moreover, as a divide-and-conquer-based algorithm, GGSpam is computationally efficient compared to machine learning-based approaches.

## 4 Bi-connected spammer group detection

Before we define the data model of the spammer groups, we make the following assumptions for group spamming activities. (1) Reviewers are required to fulfill the spamming campaign in a relatively short time period to pursue a maximum impact on target products; (2) Reviewers in the same spamming campaign either rate high scores to promote a product, or rate low scores to demote a product; (3) Each reviewer in the same campaign does not necessarily review all the target products, that is, reviewers can balance the review task by reviewing a subset of the target products. In this section, under the GGSpam computing framework, we describe the data model of a spammer group, its group spam indicators, the method to construct the reviewer graph, the divide-and-conquer-based spammer group detection algorithm, and the reordering techniques to rank detected spammer groups. We list the relevant notations and their meanings in Table 1.

### 4.1 The data model of bi-connected spammer groups

A bi-connected graph is a connected graph that is not broken into disconnected pieces by deleting any single vertex (and its incident edges). Conceptually, a bi-connected graph is a connected graph that, for each pair of node $i$ and $j$, there exist two disjoint paths between node $i$ and $j$. Bi-connected graphs are widely used in the field of networking due to its property of redundancy. As an instantiation of GGSpam, we model a spammer group as a set of review spammers who form a bi-connected graph. To begin with, we define the concept of a loose spammer group, based on which we derive the definition of the bi-connected spammer group.

**Definition 1** *Loose spammer group* A loose spammer group (or spammer group for short) $g$ is modeled as a sextet form $(R_g, P_g, V_g, S_g, SS(g), \tau)$, where $\tau$ is a user-specified time window during which a co-review action is captured; $R_g \subseteq \mathcal{R}$ is the set of review spammers (or members) in group $g$; $P_g \subseteq \mathcal{P}$ is the set of target products in group $g$, which refers to the products that are co-reviewed by at least two review spammers in $R_g$ within time window $\tau$; $V_g \subseteq \mathcal{V}$ is the set of reviews written by reviewers in $R_g$ toward products in $P_g$ during

**Table 1** Notation table

| | |
|---|---|
| $r$ | A reviewer |
| $v$ | A review |
| $p$ | A product |
| $g$ | A spammer group |
| $\mathcal{R}$ | Reviewer set |
| $\mathcal{V}$ | Review set |
| $\mathcal{P}$ | Product set |
| $R_g$ | Review spammer set of group $g$ |
| $P_g$ | Target product set of group $g$ |
| $V_g$ | Spam review set of group $g$ |
| $S_g$ | Spam indicator set of group $g$ |
| $G_g$ | Spammer group graph of group $g$ |
| $\mathrm{SS}(g)$ | Scoring function of group $g$ |
| $R_p$ | Reviewer set of product $p$ |
| $P_r$ | Product set reviewed by $r$ |
| $\varsigma_k$ | The spam suspicion of product $k$ |
| $t_i^k$ | The review time when reviewer $i$ reviews product $k$ |
| $\psi_i^k$ | The five-star rating score of reviewer $i$ for product $k$ |

co-review time window $\tau$, $V_g \subseteq R_g \times P_g$; $S_g$ is a set of spam indicators measuring the spamicity of group $g$ from different dimensions; $\mathrm{SS}(g)$ is a scoring function of $S_g$, which measures the spamicity of group $g$.

Note that $\tau$ plays an important role in GGSpam framework, as spam campaigns are highly related to time period. As we will see in Sect. 5, [1] absolutely ignores timestamps, resulting in low detection accuracy. [16] uses timestamps as metadata (priors), which takes little effect in spotting fake review(er)s.

The group members in $g$ inherently form a spammer group graph [19]. The node set of the graph is $R_g$, and for any two members in the group, their adjacent edge reflects to what extent the two reviewers collude in promoting/demoting the common target products in $P_g$.

**Definition 2** *Co-review collusiveness* Given reviewer $i$, $j \in \mathcal{R}$, if $i$ and $j$ co-review a product $k \in \mathcal{P}$, we define the collusiveness of $i$, $j$, $k$ as:

$$\mathrm{Collu}(i,j,k) = \begin{cases} 0, & |t_i^k - t_j^k| > \tau \vee |\psi_i^k - \psi_j^k| \geq 2 \\ \varsigma_k \left[ \alpha \left( 1 - \frac{|t_i^k - t_j^k|}{\tau} \right) + (1-\alpha) \left( 1 - \frac{|\psi_i^k - \psi_j^k|}{2} \right) \right], & \text{otherwise} \end{cases}$$

$$\varsigma_k = \frac{2}{1 + \mathrm{e}^{-(\mathrm{MP} - \deg(k))^\theta + 2^\theta}} - 1 \tag{1}$$

where $\alpha$ is a coefficient to balance the importance of the time difference and the rating score difference, $\varsigma_k$ is the suspicion degree of product $k$, $\deg(k)$ is the number of reviewers who reviewed product $k$, MP and $\theta$ are user-specified parameters.

Note that if reviewer $i$ and $j$ co-review product $k$ beyond time window $\tau$ or their rating score deviation is greater than 1, the co-review would not be considered, which significantly reduces the number of trivial co-reviews (co-reviews by coincidence). Figure 1 plots $\varsigma_k$ for

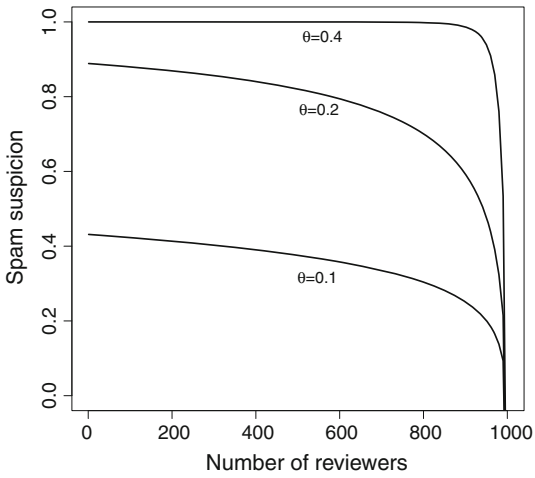



**Fig. 1** Product suspicion versus num. of reviewers of the product (MP = 1000)

$\theta = 0.1, 0.2$, and $0.4$ (MP $= 1000$). Intuitively, the larger the number of reviewers of a product, the less likely the product involves spamming.

**Definition 3** *Bi-connected spammer group graph* For a given spammer group $g$, the bi-connected spammer group graph of $g$ (if exists) is a bi-connected and weighted graph, denoted by $G_g = (R_g, E)$, where $E \subseteq R_g \times R_g$ is the edge set. For reviewer $i, j \in R_g$, the weight of edge $(i, j)$, denoted by $\omega(i, j) \in [0, 1]$ , is defined as:

$$\omega(i, j) = \frac{2}{1 + \mathrm{e}^{-\sigma(i,j)}} - 1 \tag{2}$$

where

$$\sigma(i, j) = \left[ \sum_{k \in P_i \cap P_j} \mathrm{Collu}(i, j, k) \right] \frac{|P_i \cap P_j|}{|P_i \cup P_j|} \tag{3}$$

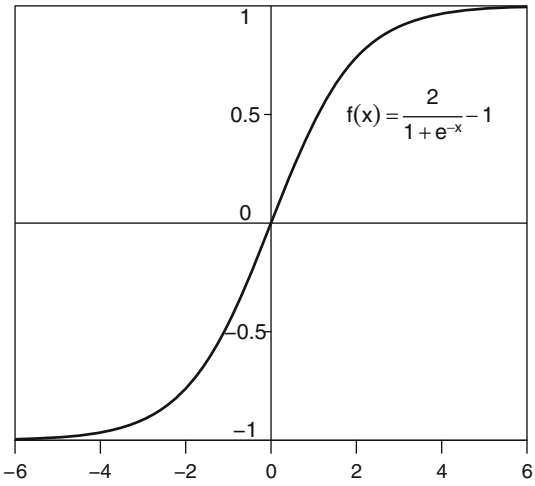
where $P_i$ and $P_j$ are the sets of products reviewed by $i$ and $j$, respectively.

The Jaccard similarity coefficient in Eq. 3 reflects the fact that collusion is more likely to happen when most of the products reviewed by the two reviewers are in common. $\omega(i, j)$ can be considered as the normalization of $\sigma(i, j)$ using the Sigmoid function which is plotted in Fig. 2.

**Definition 4** *Bi-connected spammer group* Given a spammer group $g$, if the spammer group graph of $g$ is a bi-connected spammer group graph, we call $g$ a bi-connected spammer group.

There are many advantages to model spammer groups as bi-connected graphs rather than merely connected graphs as in [19]. (1) A bi-connected component is more tightly coupled than a connected component; therefore, it can better reflect the fact that review spammers work collaboratively to fulfill the campaign task; (2) given a connected graph $G$, we can compute its bi-connected components efficiently in linear time [7]; (3) a group of reviewers forming a connected graph might seem to be normal, while its bi-connected subgraphs might be highly suspicious spammer groups; (4) although some spammer groups might be very loosely connected which results in many bi-connected subgroups, these bi-connected subgroups can be easily combined based on the nearby time window and/or similar target product set.

**Fig. 2** The Sigmoid function



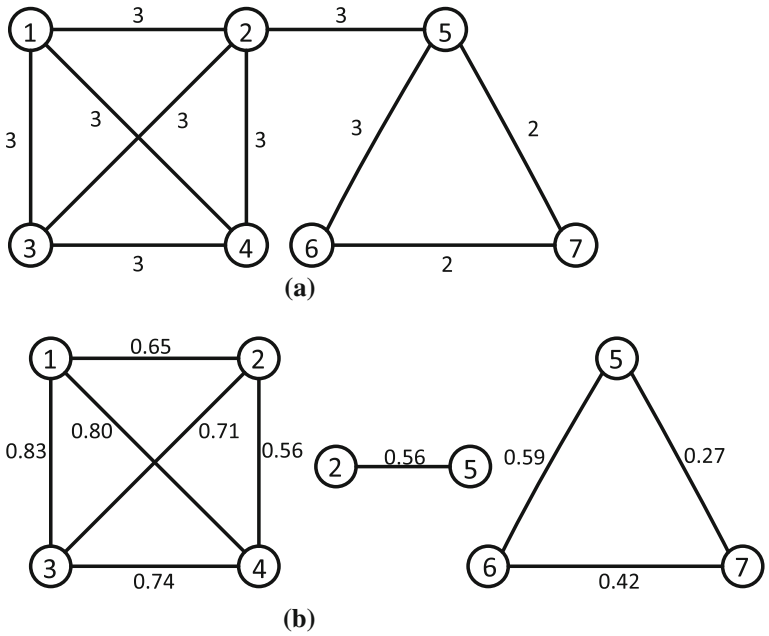




**Fig. 3** A 2-connected spammer group and its three bi-connected spammer groups. **a** A 2-connected component in GSBP. **b** Three bi-connected components

To illustrate the rationale of bi-connected spammer groups, we take a closer look at a real-world spammer group from Amazon.com detected by the method in [19][1] as show in Fig. 3a. The edge weights in Fig. 3a denote the number of commonly reviewed products by two reviewers as defined in [19], while the edge weights in Fig. 3b denote the similarity values computed by Eq. 2. We can see that this group is very sparse and the reviewers loosely

[1] http://www.amazon.com/gp/pdp/profile/AL3IL4XVAJVVR, A2K2Z8HF242WQR, A2R7ZHGHQPHLL7, ABWMFZUSJGTG7, A1XI5QKMR9NCAL, A2AOMOVZXSIQYX, A35KS5LSEVAVUD

reviewed a set of target products in a very narrow time window. By modeling spammer groups as bi-connected graphs, this group splits into 3 overlapping bi-connected spammer groups as shown in Fig. 3b, and the spamicity scores [SS($g$)] of the 3 groups are 0.70, 0.51, and 0.50, respectively. We can see that each bi-connected spammer group is more tightly coupled than the original connected component in Fig. 3a, which suggests a more suspicious spamming group. According to the human evaluation, all the 7 reviewers were identified as spammers. However, if the holistic group were not identified as a spammer group, by the bi-connected graph modeling, we would still have chance to check for spamicity at a smaller scale, whereas in [19], by raising the weight threshold from 2 to 3, the third bi-connected group will disappear. On the other hand, the 3 bi-connected spammer groups cover all the 7 reviewers in the whole spammer group and it is easy to combine them into the holistic spammer group based on the time window and the common products they review.

### 4.2 Group spam indicators

It is very important to design effective spamming indicators to measure the suspiciousness of group spamming. In [19] we proposed 8 spam indicators for group spamming behaviors. Here, we also give 8 group spam indicators on the basis of the bi-connected graph data model. One advantage of our GGSpam-based approaches is the ability to detect spammer groups of desired spam characteristics by adopting different spamming indicators. Weights can also be assigned to the spam indicators based on domain knowledge or empirical analysis. In this study, we do not use linguistic features which usually incur heavy computation overload, although these indicators are convenient to be incorporated into the detection system. Indicators marked * are identically defined in [19], whereas we still list them here for completeness. Note that we also use the penalty function defined in [19],

$$L(g) = \frac{1}{1 + \mathrm{e}^{-(|R_g| + |P_g| - 3)}} \tag{4}$$

to reduce the contingency of small-sized spammer groups. As the minimum number of reviewers is 2 and the minimum number of products is 1, $L(g) \in [0.5, 1)$.

1. *Review tightness* (*RT**) For a given spammer group $g$, the review tightness of $g$, denoted by $RT(g)$, is defined as the ratio of the number of reviews in $g$ to the cardinality of the cartesian product of the reviewer set and the product set in $g$, and multiplies $L(g)$.

$$\mathrm{RT}(g) = \frac{|V_g|}{|R_g||P_g|} L(g) \tag{5}$$

2. *Neighbor tightness* (*NT*) In a tightly coupled spammer group, the collusion relationship among reviewers intends to be stronger than those in genuine reviewer groups. Thus we define the Neighbor tightness of a group $g$ as:

$$\mathrm{NT}(g) = \frac{\sum_{i,j \in R_g} \omega(i, j)}{\binom{|R_g|}{2}} L(g) \tag{6}$$

   This indicator is a counterpart of the $NT$ indicator defined in [19] where the collusion behavior is computed only by the average value of the Jaccard similarities of the product sets of each reviewer pair.
3. *Product tightness* (*PT**) For tight spammer groups, group members concentrate on a certain number of products, and if these reviewers do not review any other products, they are most likely to be spammers. Given a spammer group $g$, the product tightness of $g$ is

defined as the ratio of the number of common products reviewed by all the members in $g$ to the number of products reviewed by all the members in $g$:

$$\mathrm{PT}(g) = \frac{|\cap_{r \in R_g} P_r|}{|\cup_{r \in R_g} P_r|} \tag{7}$$

4. *Average time window* (*TW*) Compared to randomly formed groups, spammer groups often post fake reviews in a short time period. Given a spammer group $g$, and a product $p \in P_g$, we define the time window spamicity of $p$ as:

$$\mathrm{TW}_p(g, p) = \begin{cases} 1 - \frac{\mathrm{SD}_p}{T}, & \mathrm{SD}_p \leq T \\ 0, & \mathrm{SD}_p > T \end{cases}$$

where $\mathrm{SD}_p$ is the standard deviation of review time for product $p$ reviewed by reviewers in $R_g$, $T$ is a user-specified time threshold, say, 30 days. Unlike [19], which considers the distance between the first and the last review date, we use the standard deviation of review dates to take account of the overall review time distribution. The TW indicator of $g$ is then defined as the average time window spamicity of all products in $P_g$:

$$\mathrm{TW}(g) = \mathrm{avg}_{p \in P_g} \mathrm{TW}_p(g, p) L(g) \tag{8}$$

5. *Rating variance* (*RV**) Group members tend to rate identical or similar scores. We use the same method to define the RV indicator as in [19]:

$$\mathrm{RV}(g) = 2\left(1 - \frac{1}{1 + \mathrm{e}^{-\mathrm{avg}_{p \in P_g} S^2(p,g)}}\right) L(g) \tag{9}$$

where $S^2(p, g)$ denote the variance of the rating scores of product $p$ by reviewers in $g$.

6. *Reviewer ratio* (*RR*) If the target products in $P_g$ are mainly reviewed by the reviewers in $R_g$, while reviewers not in $R_g$ are rare, then the group is more likely to be a spammer group. RR is defined as the maximum ratio of the number of reviewers in $R_g$ who review product $p$ to the number of all the reviewers of $p$, $p \in P_g$:

$$\mathrm{RR}(g) = \max_{p \in P_g} \frac{|R_{g\,p}|}{|R_p|} \tag{10}$$

7. *Multiple Review* (*MR*) Review spammers often post duplicate or similar reviews many times to attract eyeballs. Given a spammer group $g$, we define the MR indicator as the ratio of the number of reviews which involve multiple reviewing to the number of total reviews in group $g$:

$$\mathrm{MR}(g) = \frac{|\{v | v \in V_g, |(R_v, P_v)| \geq 2\}|}{|V_g|} \tag{11}$$

8. *Group size* (*GS**) Large spammer groups are more interesting than small groups because large groups are more damaging than smaller ones. On the other hand, a large portion of small groups (of size 2 or 3) are formed by accident rather than on purpose. Therefore, we take into account the group size indicator which is in favor of large spammer groups. The GS indicator is defined as

$$\mathrm{GS}(g) = \frac{1}{1 + \mathrm{e}^{-(|R_g| - 3)}} \tag{12}$$

We do not use the *Early Review* (*ER*) indicator used in [19], because we observed that the ER indicator often correlates to other indicators, e.g., PT, RR. Meanwhile, we introduce a new indicator MR to capture the fact that spammers often review a product many times to attract eyeballs.

### 4.3 The GSBC algorithm

In [19], we give a divide-and-conquer-based algorithm (GSBP) to mine $k$-connected spammer groups. There the spammer groups are obtained by incrementing the edge weight threshold $k$ by one, $k \geq 1$, to weaken the connectivity of a graph so that strongly connected subgraphs appear. In this study, we introduce notably different data model for the spammer groups: (1) Spammer groups are modeled as bi-connected rather than loosely connected components; (2) The edge weights become real numbers ranged in [0, 1] rather than integers. Therefore, GSBP is no longer applicable to our proposed bi-connected spammer groups. In this section, we give a new approach to mining bi-connected spammer groups under the GGSPAM computing framework. Specifically, we first design an efficient algorithm to compute the global reviewer graph, and then, we traverse the reviewer graph to find all the bi-connected components in each connected component of the reviewer graph. For large and/or unsuspicious bi-connected graphs, we use minimum cut algorithm to split them into smaller connected graphs so that they can be further investigated for spam detection recursively.

Algorithm 1 (GSBC) takes as input a bipartite graph $\mathcal{B}$ which consists of reviewer nodes and product nodes, the time window $\tau$ and a edge weight threshold $\delta$. It first invokes Algorithm 2 (ConstructReviewerGraph) to construct a global reviewer graph, namely $G = (\mathcal{R}, E)$, where $E$ is the edge set. For each edge $e \in E, \omega(e) \geq \delta$. Then for every connected component $g$ in $G$, Algorithm 3 (FindGroups) is invoked to mine all the bi-connected spammer groups.

Algorithm 2 (ConstructReviewerGraph) is an efficient algorithm for computing the reviewer graph. It checks the common reviews toward product $k$ between reviewer $i$ and reviewer $j$. The edge weight between $i$ and $j$ is computed according to Eqs. 1–3. Edges with weight $<\delta$ are removed from the graph to reduce the computation complexity in mining bi-connected spammer groups. The time complexity of Algorithm 2 is $O(|\mathcal{V}|)$, because each review (an edge in $\mathcal{B}$) is visited exactly twice.

Algorithm 3 (FindGroups) is a divide-and-conquer-based algorithm. For each connected component $c$ in reviewer graph $G$, it searches for all the bi-connected components $\{b\}$ in $c$. If

**Algorithm 1** *Group Spam detection via Bi-Connected graphs (GSBC)*

**Input:**

$\mathcal{B}$: bipartite graph representing reviewers, products and reviews;
$\tau$: review time window
$\delta$: Similarity threshold
$MINSPAM$: minimum spam threshold for a group
$MAXSIZE$: maximum size of a group

**Output:**

Bi-connected spammer groups satisfying $\text{SS}(g) \geq MINSPAM$

**Description:**

1: $G$ = ConstructReviewerGraph($\mathcal{B}$, $\tau$, $\delta$); {Construct the global reviewer graph}
2: **for** each connected component $c$ in $G$ **do**
3: FindGroups($c$);// finds all bi-connected spammer groups
4: **end for**

**Algorithm 2** *ConstructReviewerGraph($\mathcal{B}$, $\tau$, $\delta$)*

**Output:**

Global reviewer graph with edge weight $\geq \delta$

**Description:**

Set $\omega(i, j) = 0$ for any $i, j \in \mathcal{R}$;
**for** each reviewer $i$ in $\mathcal{R}$ **do**
  **for** each product $k$ in $P_i$ **do**
    **for** each reviewer $j$ in $R_k$ **do**
      $\omega(i, j) = \omega(i, j) + Collu(i, j, k)$
    **end for**
  **end for**
**end for**
**for** each $i, j \in \mathcal{R}$ **do**
  Set $\omega(i, j) = \omega(i, j)\frac{|P_i \cap P_j|}{|P_i \cup P_j|}$
  Set $\omega(i, j) = \frac{2}{1+e^{-\omega(i,j)}} - 1$
  **if** $\omega(i, j) < \delta$ **then**
    $\omega(i, j) = 0$
  **end if**
**end for**
**return** reviewer graph $(\mathcal{R}, E)$

**Algorithm 3** *FindGroups($c$)*

**Input:**

$c$: a connected component;

**Output:**

Bi-connected spammer groups satisfying $SS(g) \geq MINSPAM$ in $c$

**Description:**

**for** each bi-connected component $b$ in $c$ **do**
  **if** $b.size > MAXSIZE$ **then**
    $c_1, c_2$=MinCut($b$);
    FindGroups($c_1$);
    FindGroups($c_2$);
  **else if** $SS(b) \geq MINSPAM$ **then**
    Output $b$;
  **else if** $b.size > 2$ **then**
    $c_1, c_2$=MinCut($b$);
    FindGroups($c_1$)
    FindGroups($c_2$)
  **end if**
**end for**

the size (number of reviewers) of the bi-connected component $b$ is below the given threshold MAXSIZE, then the group spam indicators are computed and if SS($b$) is greater or equal to the minimum spamicity threshold (MINSPAM), then $b$ is output as a bi-connected spammer group. Otherwise, if the size of $b$ is greater than MAXSIZE, or SS($b$) is below MINSPAM, then $b$ is divided into two connected component, $c_1$ and $c_2$, by the minimum cut algorithm. Then $c_1$ and $c_2$ are further processed by Algorithm 3 recursively.

Searching for bi-connected graphs in a connected graph takes linear time in the number of the nodes of the graph [7], that is, $O(|\mathcal{R}|)$. Bi-connected groups whose $\mathrm{SS}(g) < \mathrm{MINSPAM}$, however, have to be split into two connected subgroups to further investigate their spamicity. To safely split a large bi-connected graph into smaller connected components, we employ the minimum cut algorithm which split a connected graph by removing some edges such that the sum of their weights is minimum. The time complexity of the minimum cut algorithm is $O(|V||E|+|V|^2 \log |V|)$ by using a priority queue [17], where $V$ and $E$ are the vertex set and the edge set of the graph, respectively. Fortunately, most of the bi-connected graphs have less than MAXSIZE nodes in real-world review dataset. As such, the overall time complexity of Algorithm 3 is near $O(|\mathcal{R}| \log |\mathcal{R}|)$. The final running time is severely affected by the number of edges in $G$. Nonetheless, we can always control the number of edges (i.e., the outliers) by specifying a sufficiently large $\delta$, a predefined parameter.

### 4.4 Spammer group ranking

The last step in GGSpam, which can be viewed as a post-processing phase, is to rank the spammer groups detected by a divide-and-conquer-based algorithm, e.g., GSBC. The goal of this step is to improve the precision of the algorithm by analyzing various spam indicators. Besides the 8 group spam indicators, other spam indicators, e.g., linguistic features of review content proposed in [10] shall also be useful to determine the spamicity of the groups.

The most simple yet effective way to rank the detected bi-connected spammer groups is by averaging the 8 group spam indicator values. However, the 8 group spam indicators have different capabilities to discriminate the spam/non-spam behaviors of a spammer group. Naturally, we can use linear regression to determine the weights of the spam indicators. To train a regression model, we need to know the labels of the detected spammer groups. However, group spam labels are often hard to obtain. In such cases, pseudo-supervised method can be used, i.e., treating the top N groups as fake review groups, and randomly formed groups as genuine review groups.

## 5 Experimental study

### 5.1 The datasets

We use 5 real-world datasets to evaluate our GSBC approach, two are from Amazon.com without ground-truth, three are from Yelp.com with near-ground-truth. Table 2 shows the summary statistics of the five datasets. AmazonBooks contains book reviews extracted from the Amazon dataset[2] crawled in 2006, comprising reviews from 1996 to 2006, which was also used in [8,9,11–13,19]. Note that this dataset involves multiple reviewing. Since spam mechanisms have undoubtedly advanced a lot since 2006, we also used a recent Amazon dataset, AmazonCDs, which comprises CDs and Vinyl reviews from 2012 to 2014.[3] This dataset contains rating data only and does not involve multiple reviewing. The YelpChi dataset contains reviews for restaurants and hotels in the Chicago area and was collected by [14]. YelpNYC and YelpZip are another two Yelp datasets collected by [16] which contain hotel/restaurant reviews from NYC and a number of areas with continuous zipcode started from NYC, respectively. The three Yelp datasets contain both recommended and filtered

[2] http://liu.cs.uic.edu/download/data/.

[3] http://jmcauley.ucsd.edu/data/amazon/.

**Table 2** Review dataset statistics

| Dataset | #Reviews | #Reviewers | #Products | Time span | Labeled |
|---|---|---|---|---|---|
| AmazonBooks | 1,158,930 | 145,942 | 197,038 | 1996.5–2006.5 | No |
| AmazonCDs | 972,105 | 536,264 | 118,122 | 2012.1–2014.7 | No |
| YelpChi | 67,395 | 38,063 | 201 | 2004.10–2012.10 | Yes |
| YelpNYC | 359,052 | 160,225 | 923 | 2004.10–2015.1 | Yes |
| YelpZip | 608,598 | 260,277 | 5,044 | 2004.10–2015.1 | Yes |

reviews labeled by the Yelp anti-fraud filtering algorithm, and thus, they can be treated as near-ground-truth datasets [16]. The Yelp datasets do not involve multiple reviewing.

### 5.2 Compared baselines

Since GSBP is also a GGSpam instantiation, we take GSBP as a baseline to compare with. We also introduced SCAN [24], a graph-based clustering algorithm, to perform clustering on our global reviewer graph in order to generate dense subgraphs. These dense clusters can also be viewed as suspicious spammer groups. Although SCAN also exploits our global reviewer graph model, it differs from GGSpam framework in that it does not consider the spamicity of each cluster during the clustering process. We also compare GSBC with FraudEagle [1] and SpEagle [16], which also use the three Yelp datasets to evaluate their performance. Essentially, FraudEagle and SpEagle do not generate spammer groups. Instead, they only rank reviewers or reviews. Note that FraudEagle and SpEagle are probabilistic graphic model-based solutions, which exploit Loop Belief Propogation (LBP) to inference the spamicity of review(er)s. In [16], a semi-supervised version, namely SpEagle$^+$, is proposed, which can greatly improve the detection precision. However, our method is absolutely unsupervised, and labeling fake reviews is often impossible by merely reading the review text, so we do not compare to SpEagle$^+$ in this study.

### 5.3 Performance on unlabeled datasets

We evaluate GSBC against GSBP and SCAN on the two unlabeled Amazon datasets. As we optimized the group spam indicators defined in [19], for fairness and consistency, we tried our best to use the same indicator set in GSBP as those in GSBC. That is, we do not use the ER (Early review) indicator, and instead, we use MR (Multiple reviewing) defined in this paper. The TW indictor is also computed in the same way as in this paper.

Since AmazonBooks and AmazonCDs are unlabeled datasets, it is difficult to directly evaluate the effectiveness of various spam detection approaches. Borrowing the idea in [4] which evaluates the spamicity of detected spammer groups via spam indicators, we compared the *cumulative distribution function* (CDF) curves of various spam indicators for the spammer groups detected by GSBC, GSBP and SCAN. The CDF curve, first proposed in [12], is often used to compare the distribution of a series of spam indicators.

For each dataset, we use GSBC to generate 500+ groups. The parameters used in GSBC as well as the number of groups generated and the time elapsed for each dataset are shown in Table 3. We adjust the parameters in GSBP and SCAN so that the number of groups generated is comparable to the number of groups return by GSBC. For fairness, we, respectively, fetched the top 500 spammer groups detected by GSBC, GSBP, and SCAN. Then we compute their 8 spam indicators in the way defined in Sect. 4.2. Figures 4 and 5 shows the CDF

**Table 3** GSBC parameter setting, #groups, and time elapsed (MAXSIZE = 50, $\theta = 0.2$, $\alpha = 0.5$, for all datasets)

| Dataset | $\tau$ | $\delta$ | MP | MINSPAM | #Groups | Time (s) |
|---|---|---|---|---|---|---|
| AmazonBooks | 30 | 0.1 | 1,000 | 0.49 | 545 | 185 |
| AmazonCDs | 15 | 0.4 | 1,000 | 0.65 | 561 | 242 |
| YelpChi | 10 | 0.3 | 10,000 | 0.56 | 530 | 42 |
| YelpNYC | 10 | 0.4 | 10,000 | 0.58 | 628 | 676 |
| YelpZip | 10 | 0.4 | 10,000 | 0.59 | 551 | 850 |

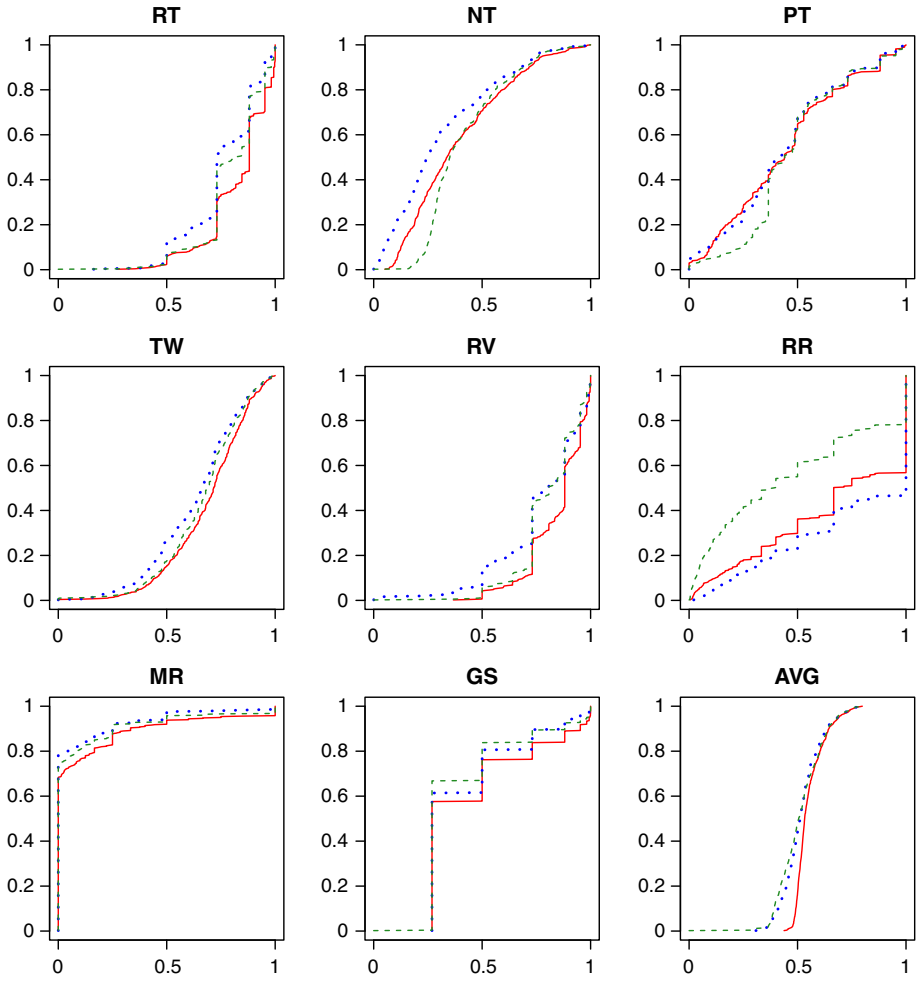



**Fig. 4** Top-500-groups CDF comparison of GSBC (*red* and *solid*), GSBP (*blue* and *dotted*) and SCAN (*green* and *dashed*) on AmazonBooks. AVG is the average of all 8 group spam indicators. The same below (color figure online)

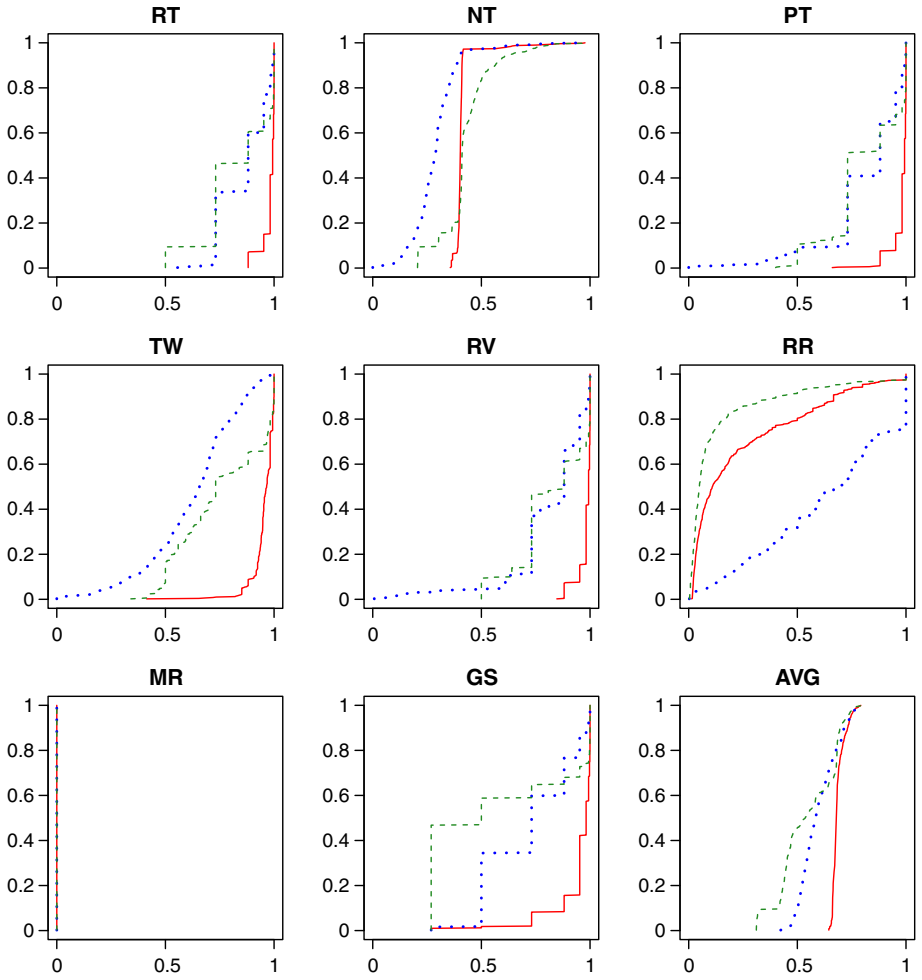



**Fig. 5** Top-500-groups CDF comparison of GSBC (*red* and *solid*), GSBP (*blue* and *dotted*) and SCAN (*green* and *dashed*) on AmazonCDs (color figure online)

curves of the 8 indicators and their average value (AVG) for spammer groups detected by GSBC (red and solid lines), GSBP (blue and dotted) and SCAN (green and dashed) on dataset AmazonBooks and AmazonCDs, respectively. The closer the curve is to the vertical axis, the smaller their indicator values will be. We can see that, for most of the indicators, GSBC generates higher scores than GSBP and SCAN. For AVG indicator, GSBC always outperforms GSBP and SCAN by a large margin, while the distinction between GSBP and SCAN is not significant. Note that AmazonCDs does not involve multiple review, so its MR values are all zeros. To our surprise, GSBC performs extremely well on AmazonCDs dataset. This is because AmazonCDs contains more up-to-date review data (2012–2014) than those in AmazonBooks (1996–2006). Definitely, the number of internet users increased exponentially in recent years, and co-reviewing a product is becoming prevalent; thus, more group spam phenomena emerge. Also, the fact that SCAN gains comparable performance to GSBP indicates that the way we model the collusiveness between a pair of reviewers through Eq. 2 is effective.

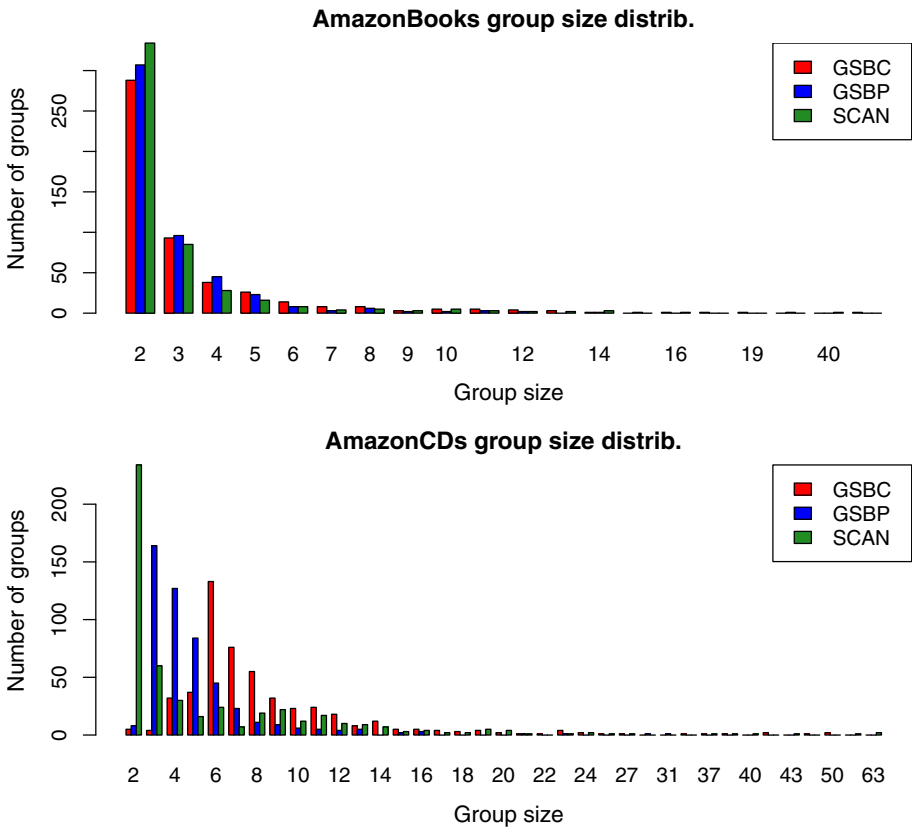



**Fig. 6** Group size comparison on Amazon datasets (Top 500 groups)

We notice that for both Amazon datasets, RR values in GSBP are significantly higher than those in GSBC. Deep spammer group analysis reveals that, the reason why RR behaves differently from other indicators is mainly because GSBP models the reviewer similarity (collusiveness) as the number of co-reviewed products, while GSBC models the reviewer similarity by both review time and rating scores of the co-reviewed products, thus, reviewers who co-review a product within time window $\tau$ but the rating score difference is relatively high will be filtered out by GSBC.

Figure 6 depicts the number of spammer groups versus group size for GSBC, GSBP and SCAN on AmazonBooks and AmazonCDs, respectively. We can see that, in general, the number of groups decreases as the group size increases. For AmazonBooks, each method returns a large number of groups with only 2 reviewers. However, GSBC tends to generate larger groups than GSBP and SCAN. For AmazonCDs which involves larger number of co-reviews, GSBC returns significantly smaller number of groups with only 2 or 3 reviewers than GSBP and SCAN. As a result, GSBC can spot more individual group spammers than GSBP and SCAN.

### 5.4 Performance on labeled datasets

For labeled Yelp datasets, we not only give CDF comparison against GSBP and SCAN, but also give precision comparison against GSBP, SCAN, FRAUDEAGLE, and SPEAGLE. Because

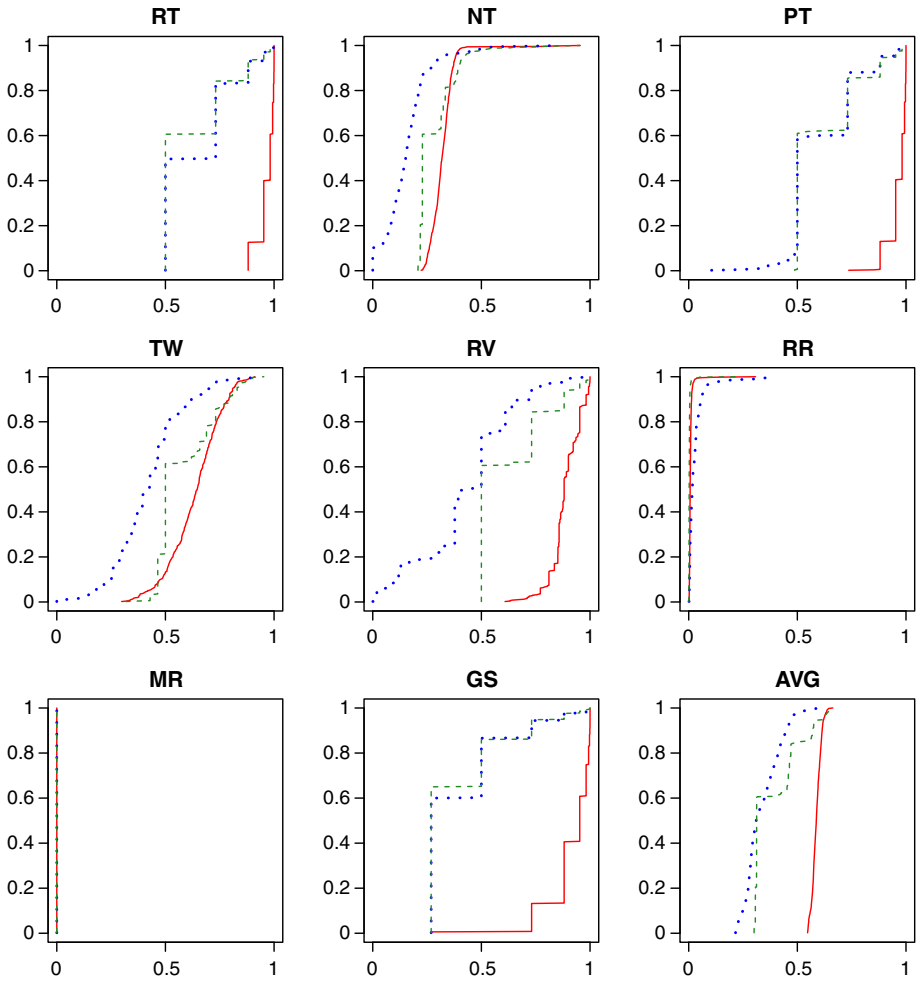



**Fig. 7** Top-500-groups CDF comparison of GSBC (*red* and *solid*), GSBP (*blue* and *dotted*) and SCAN (*green* and *dashed*) on YelpChi (color figure online)

hotel review dataset involves less number of products (hotels), which leads to considerably large number of co-reviews, we set MP to 10,000 and $\tau$ to 10 as shown in Table 3. Figures 7, 8, and 9 depict the CDF comparison on YelpChi, YelpNYC and YelpZip, respectively. We can see that on Yelp datasets, GSBC performs even better than on Amazon datasets. The gap between the curves of GSBC and GSBP or SCAN gets notably wider. This suggests GSBC is more suitable for hotel/restaurant reviews, which involve considerably less number of items (products) and more co-reviews.

Figure 10 depicts the group size on YelpChi, YelpNYC, and YelpZip, respectively. Again, GSBC generates least number of groups of size 2 or 3. As a result, GSBC can produce more individual group spammers than GSBP and SCAN.

Since Yelp datasets contain near-ground-truth, that is, each review in a Yelp dataset is labeled as either fake (filtered) or genuine (recommended), we can evaluate the precision of GSBC by checking each review(er) in the detected spammer groups. We take a reviewer as

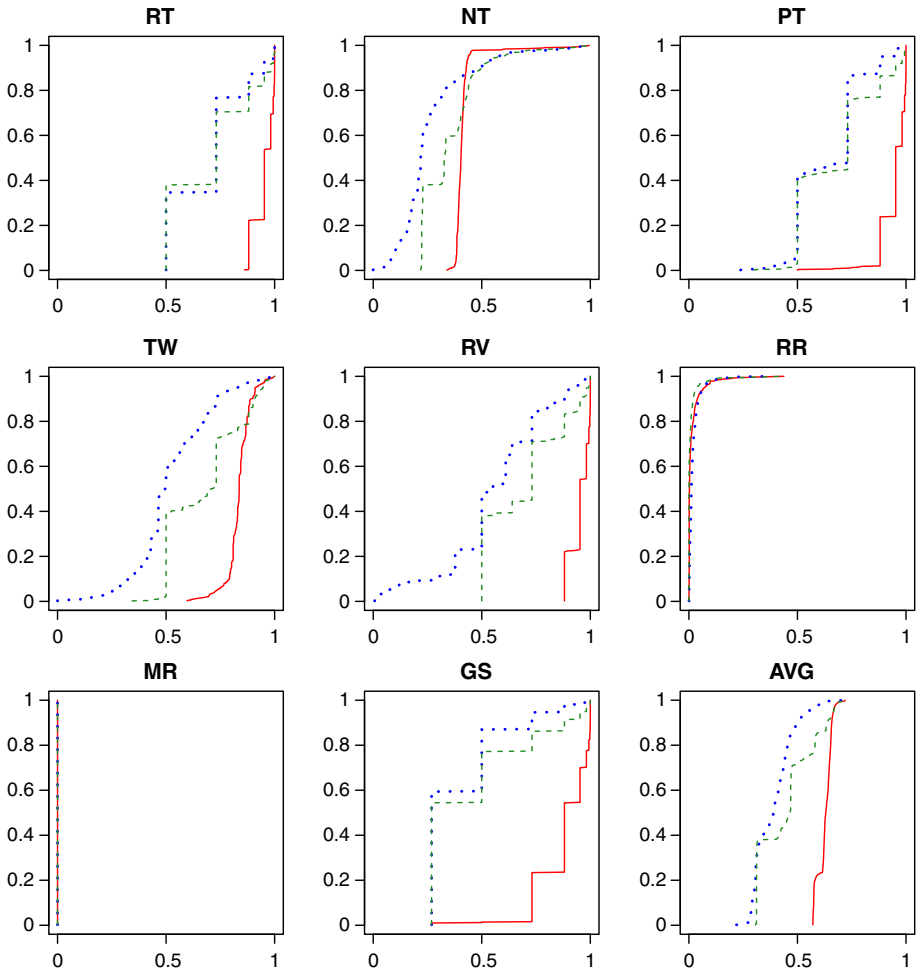



**Fig. 8** Top-500-groups CDF comparison of GSBC (*red* and *solid*), GSBP (*blue* and *dotted*) and SCAN (*green* and *dashed*) on YelpNYC (color figure online)

fake if and only if she/he has written at least 1 fake review in the group. Therefore, we can check the top-ranked review(er)s according to the detected top 500 groups they belong to. In [16], the top 1000 reviews/reviewers for each dataset by FRAUDEAGLE and SPEAGLE were reported. Figure 11 shows the reviewer precision and review precision at top k review(er)s ($k \leq 1000$) for GSBC and other four baselines on Yelp datasets. We can see that the precision decreases as the number of review(er)s gets larger, except for the reviewer precision of FRAUDEAGLE on YelpNYC dataset, which weirdly goes up as more reviewers get in. In general, GSBC outperforms all the 4 baselines, except for SPEAGLE on YelpChi dataset. It is worth noting that GSBC outperforms GSBP for both reviewer precision and review precision on all the three Yelp datasets. To our surprise, SCAN works almost as well as GSBC, which further validate the effectiveness of our modeling of reviewer relationship by Eq. 2.

Figure 12 shows the precision and F1 score at top $k$ groups for Yelp datasets. The precision at top $k$ groups is defined as the ratio of the number of fake reviews in all the top $k$ groups and the number of reviews contained in the top $k$ groups. The F1 score considers both the

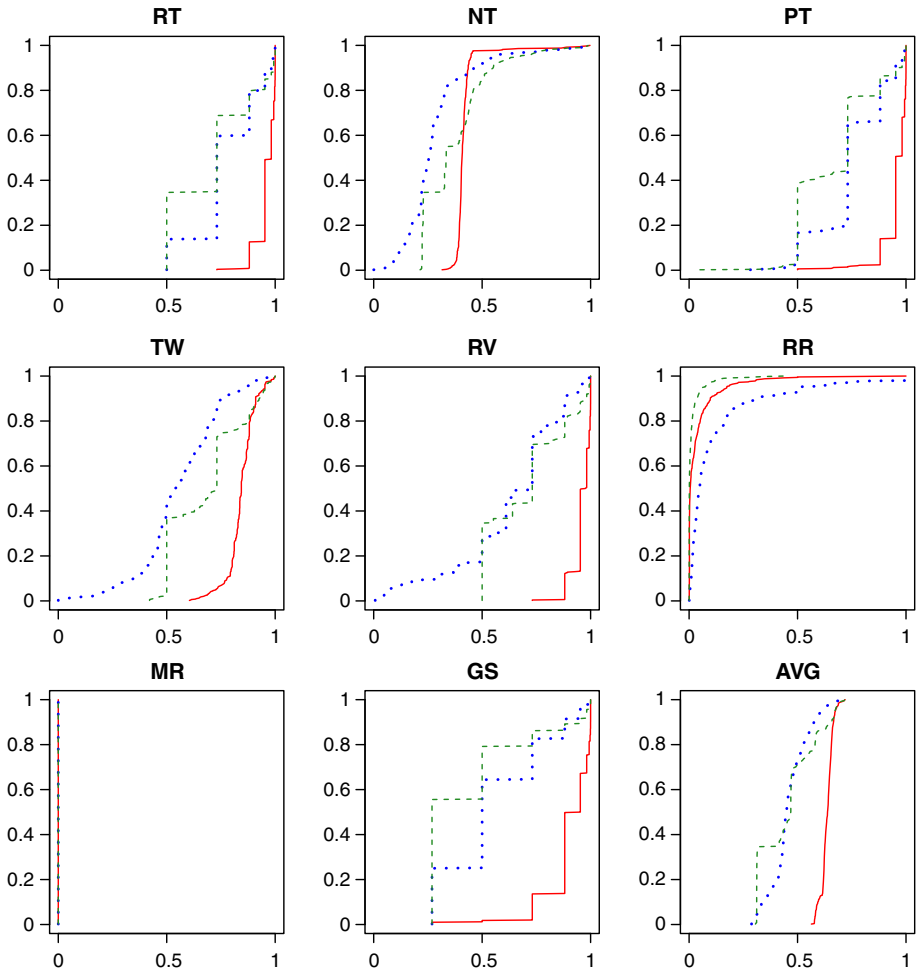



**Fig. 9** Top-500-groups CDF comparison of GSBC (*red* and *solid*), GSBP (*blue* and *dotted*) and SCAN (*green* and *dashed*) on YelpZip (color figure online)

precision and recall of reviews. For the precision on YelpChi and YelpNYC, GSBC and SCAN overwhelmingly outperform GSBP. For the precision on YelpZip, GSBC and SCAN only perform better than GSBP for the top 70 groups, and after that, GSBP outperforms GSBC and SCAN. For all Yelp datasets, GSBC always has a higher F1-score than GSBP and SCAN due to the fact that GSBC intends to generate more fake review(er)s than GSBP and SCAN. This suggests that GSBC can generate higher quality of spammer groups than GSBP and SCAN.

### 5.5 Impact of parameters

There are several user-specified parameters which affect the number and the composition of the resulting spammer groups. Here we investigate the impact of these parameters on YelpNYC dataset.

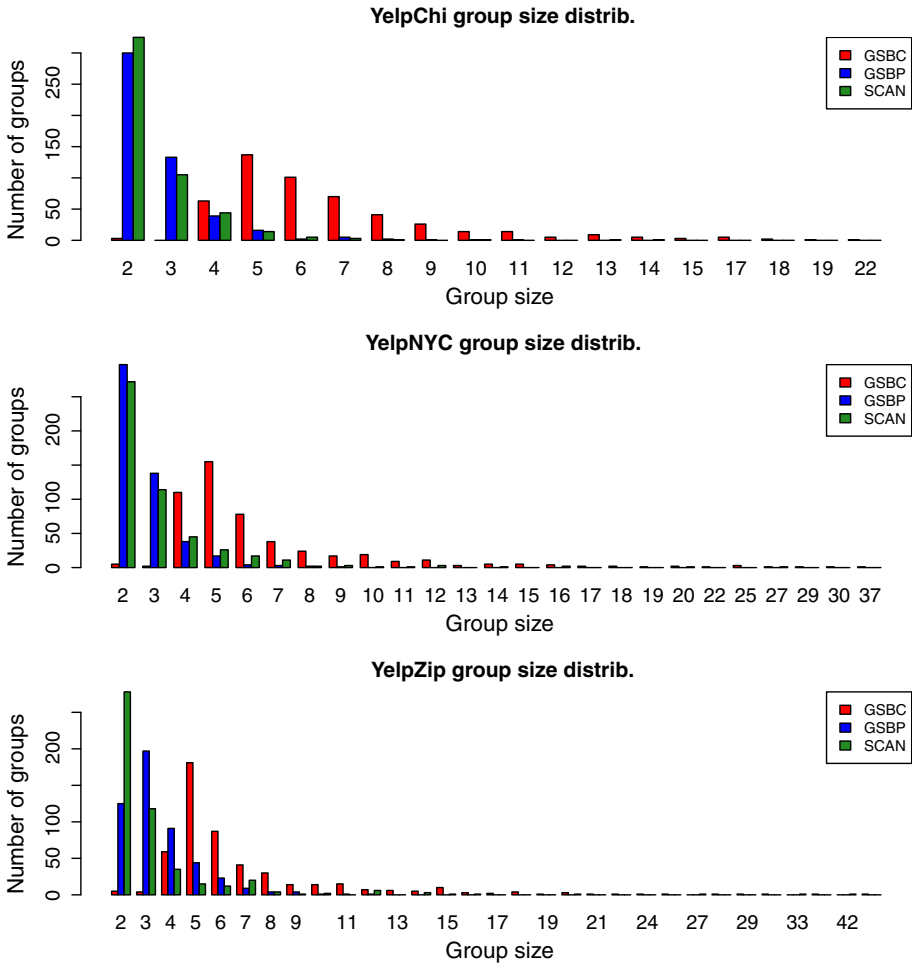



**Fig. 10** Group size comparison on Yelp datasets (Top 500 groups)

Parameter $\delta$, which is used to filter out those small collusion values between reviewers, determines the topological structure of the reviewer graph. We set $\delta$ to 0.36, 0.38, 0.40, and fix $\tau = 10$, MINSPAM $= 0.58$, MAXSIZE $= 50$. For different $\delta$ values, Table 4 shows the number of spammer groups, number of edges (weights), the running time elapsed, and the Jaccard similarity and the number of common reviewers for top 100, 200, 400 spammer groups. We use $\delta = 0.4, \tau = 10$, MINSPAM $= 0.58$, MAXSIZE $= 50$ as the baseline which is marked * in Table 4. For two spammer group sets of the top $k$ spammer groups selected from the baseline and the setting with $\delta = 0.36$ or 0.38, the Jaccard similarity between the two sets is defined as the ratio of the number of common reviewers to the cardinality of the union of the two sets. From the table we can see that lower $\delta$ value generates more edges in the reviewer graph, and thus, more spammer groups are found, and more running time is required. The Jaccard similarities decrease as $\delta$ decreases. As the number of groups increases, more common reviewers are found.

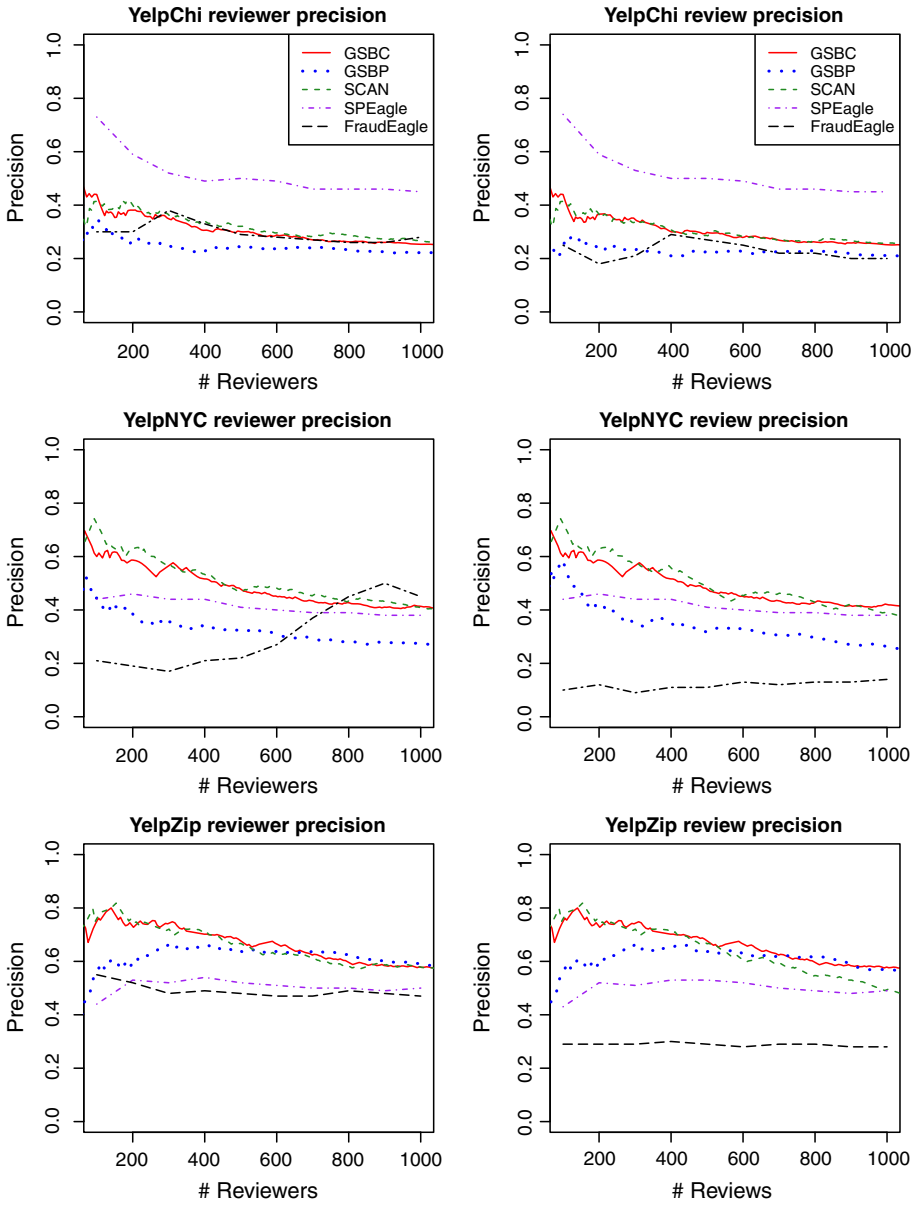



**Fig. 11** Precision comparison for top-ranked review(er)s on Yelp datasets

Similarly, to study the impact of $\tau$, we fix $\delta = 0.4$, MINSPAM $= 0.58$, MAXSIZE $= 50$, and set $\tau$ to 10, 15, 20 days. From Table 5 we can see that the number of spammer groups increases as $\tau$ increases. Although the number of edges increases, the time required only increases slightly. The Jaccard similarities decrease as $\tau$ increases. Tables 4 and 5 reveal that both $\delta$ and $\tau$ influence the resulting spammer groups greatly.

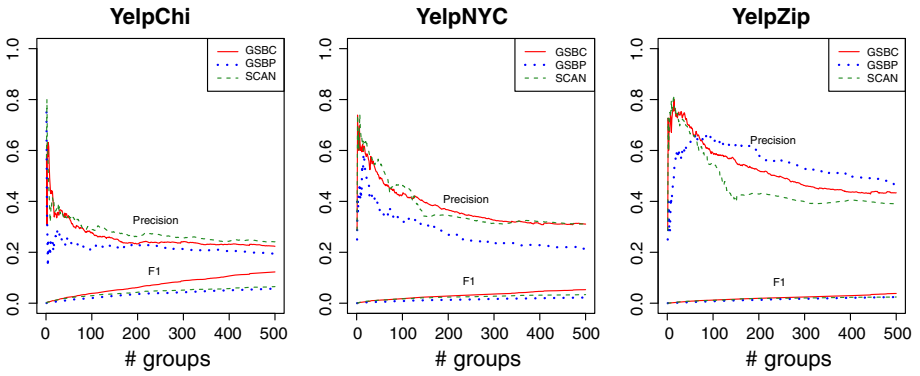



**Fig. 12** Review precision and F1 @ top k groups on Yelp datasets

**Table 4** Spammer group info w.r.t. $\delta$ ($\tau = 10$, MINSPAM $= 0.58$, MAXSIZE $= 50$). Column marked * denotes the setting of the baseline

| Type | $\delta = 0.36$ | $\delta = 0.38$ | $\delta = 0.40$* |
|---|---|---|---|
| Num. of groups | 943 | 773 | 628 |
| Num. of weights | 44389 | 35325 | 26232 |
| Running time (s) | 728 | 725 | 676 |
| J. sim./#common reviewers (Top 100) | 0.42/463 | 0.56/571 | 1.00/792 |
| J. sim./#common reviewers (Top 200) | 0.38/801 | 0.53/1006 | 1.00/1471 |
| J. sim./#common reviewers (Top 400) | 0.43/1710 | 0.56/2025 | 1.00/2868 |

**Table 5** Spammer group info w.r.t. $\tau$ ($\delta = 0.4$, MINSPAM $= 0.58$, MAXSIZE $= 50$). Column marked * denotes the setting of the baseline

| Type | $\tau = 10*$ | $\tau = 15$ | $\tau = 20$ |
|---|---|---|---|
| Num. of groups | 628 | 1124 | 1448 |
| Num. of weights | 26232 | 41941 | 51571 |
| Running time (s) | 676 | 720 | 727 |
| J. sim./#common reviewers (Top 100) | 1.00/792 | 0.48/520 | 0.37/435 |
| J. sim./#common reviewers (Top 200) | 1.00/1471 | 0.44/903 | 0.33/751 |
| J. sim./#common reviewers (Top 400) | 1.00/2868 | 0.49/1882 | 0.34/1471 |

From Table 6 we can see that MINSPAM significantly affects the number of groups detected. As expected, lower MINSPAM results in more groups. The running time is almost the same, and the Jaccard similarities are extremely high for top 100, 200, and 400 groups. We can see that MINSPAM severely impacts the number of groups generated, but it does not influence the composition of top ranked groups too much. Intuitively, lower MINSPAM will gain higher recall but lower precision.

**Table 6** Spammer group info w.r.t. MINSPAM ($\delta = 0.4$, $\tau = 10$, MAXSIZE $= 50$). Column marked * denotes the setting of the baseline

| Type | M.S. = 0.58* | M.S. = 0.50 | M.S. = 0.42 |
|---|---|---|---|
| Num. of groups | 628 | 1000 | 2434 |
| Num. of weights | 26232 | 26232 | 26232 |
| Running time (s) | 676 | 702 | 694 |
| J. sim./#common reviewers (Top 100) | 1.00/792 | 1.00/792 | 1.00/792 |
| J. sim./#common reviewers (Top 200) | 1.00/1471 | 0.99/1461 | 0.99/1461 |
| J. sim./#common reviewers (Top 400) | 1.00/2868 | 0.98/2828 | 0.98/2828 |

## 6 Conclusion

Online review spamming has been increasingly becoming a real threat to the online rating systems, and detecting group spamming activities is a critical problem to protect online customers. In this paper, we propose a general computing framework for detecting online review spammer groups, which recursively breaks the whole reviewer graph into small suspicious reviewer groups. Moreover, we propose a novel instantiation of the proposed computing framework, which models review spammer groups as bi-connected graphs and splits large bi-connected graphs by minimum cut algorithm. A number of group spam indicators are designed to evaluate the spamicity of a spammer group. Experimental study on a real-world datasets against several state-of-the-art approaches verifies the effectiveness and efficiency of our approach. Future work includes seeking new effective group spam features to further improve the precision and recall, and other possible instantiations of GGSPAM to detect various spammer groups in different domains.

**Acknowledgements** We are very grateful to Dr. Shebuti Rayana (Stony Brook University) and Prof. Julian McAuley (UCSD) for sharing their high-quality datasets used in this study. We also thank the anonymous reviewers for their insightful comments.

## References

1. Akoglu L, Chandy R, Faloutsos C (2013) Opinion fraud detection in online reviews by network effects. In: Proceedings of the seventh international conference on weblogs and social media, ICWSM, Cambridge, Massachusetts, 8-11 July 2013
2. Akoglu L, Tong H, Koutra D (2015) Graph based anomaly detection and description: a survey. Data Min Knowl Discov 29(3):626–688
3. Allahbakhsh M, Ignjatovic A, Benatallah B, Beheshti S, Bertino E, Foo N (2013) Collusion detection in online rating systems. In: Web technologies and applications (Lecture notes in computer science), vol 7808. Springer, Berlin, Heidelberg, pp 196–207
4. Choo E, Yu T, Chi M (2015) Detecting opinion spammer groups through community discovery and sentiment analysis. In: Data and applications security and privacy XXIX—29th annual IFIP WG 11.3 working conference, DBSec 2015, Fairfax, 13–15 July 2015, pp 170–187
5. Fayazi A, Lee K, Caverlee J, Squicciarini A (2015) Uncovering crowdsourced manipulation of online reviews. In: Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval, Santiago, 9–13 August 2015, pp 233–242
6. Fei G, Mukherjee A, Liu B, Hsu M, Castellanos M, Ghosh R (2013) Exploiting burstiness in reviews for review spammer detection. In: 7th International AAAI conference on weblogs and social media
7. Hopcroft J, Tarjan R (1973) Efficient algorithms for graph manipulation [H] (algorithm 447). Commun ACM 16(6):372–378

8. Jindal N, Liu B (2008) Opinion spam and analysis. In: Proceedings of the 2008 international conference on web search and data mining, ACM, New York, pp 219–230
9. Jindal N, Liu B, Lim E (2010) Finding unusual review patterns using unexpected rules. In: Proceedings of the 19th ACM international conference on information and knowledge management , CIKM '10, New York, pp 1549–1552
10. Li J, Ott M, Cardie C, Hovy E (2014) Towards a general rule for identifying deceptive opinion spam. In: Proceedings of the 52nd annual meeting of the association for computational linguistics (Vol 1: Long papers), Baltimore, pp 1566–1576
11. Lim E, Nguyen V, Jindal N, Liu B, Lauw H (2010) Detecting product review spammers using rating behaviors. In: Proceedings of the 19th ACM international conference on information and knowledge management, CIKM '10, New York, pp 939–948
12. Mukherjee A, Liu B, Glance N (2012) Spotting fake reviewer groups in consumer reviews. In: Proceedings of the 21st international conference on world wide web, ACM, New York, pp 191–200
13. Mukherjee A, Kumar A, Liu B, Wang J, Hsu M, Castellanos M, Ghosh R. (2013) Spotting opinion spammers using behavioral footprints. In: Proceedings of the 19th ACM SIGKDD international conference on knowledge discovery and data mining, New York, pp 632–640
14. Mukherjee A, Venkataraman V, Liu B, Glance N (2013) What yelp fake review filter might be doing? In: Proceedings of the 7th international conference on weblogs and social media, ICWSM, Cambridge, Massachusetts, 8–11 July 2013
15. Ott M, Choi Y, Cardie C, Hancock J (2011) Finding deceptive opinion spam by any stretch of the imagination. In: Proceedings of the 49th annual meeting of the association for computational linguistics: human language technologies, vol 1, Stroudsburg, pp 309–319
16. Rayana S, Akoglu L (2015) Collective opinion spam detection: bridging review networks and metadata. In: Proceedings of the 21st ACM SIGKDD international conference on knowledge discovery and data mining, Sydney, 10–13 Aug 2015, pp 985–994
17. Stoer M, Wagner F (1997) A simple min-cut algorithm. J ACM 44(4):585–591
18. Wang G, Xie S, Liu B, Yu P (2011) Review graph based online store review spammer detection. ICDM 2011:1242–1247
19. Wang Z, Hou T, Song D, Li Z, Kong T (2016) Detecting review spammer groups via bipartite graph projection. Comput J 59(6):861–874
20. Xie S, Wang G, Lin S, Yu P (2012) Review spam detection via time series pattern discovery. In: Proceedings of the 21st international conference companion on world wide web, New York, pp 635–636
21. Xu C, Zhang J (2015) Combating product review spam campaigns via multiple heterogeneous pairwise features. In: Proceedings of the 2015 SIAM international conference on data mining, Vancouver, April 30–May 2 2015, pp 172–180
22. Xu C, Zhang J (2015) Towards collusive fraud detection in online reviews. In: 2015 IEEE international conference on data mining, ICDM Atlantic City, 14–17 Nov 2015, pp 1051–1056
23. Xu C, Zhang J, Chang K, Long C (2013) Uncovering collusive spammers in chinese review websites. In: Proceedings of the 22nd ACM international conference on conference on information and knowledge management, ACM, New York, pp 979–988
24. Xu X, Yuruk N, Feng Z, Schweiger T (2007) SCAN: a structural clustering algorithm for networks. In: Proceedings of the 13th ACM SIGKDD international conference on knowledge discovery and data mining, San Jose, 12–25 Aug 2007, pp 824–833
25. Ye J, Akoglu L (2015) Discovering opinion spammer groups by network footprints. In: Machine learning and knowledge discovery in databases (Lecture notes in computer science), vol 9284. Springer International Publishing, pp 267–282

**Zhuo Wang** received his B.S. degree in Computer Software from Xidian University, Xi'an, China, in 1992 and M.S. degree in Computer Software and Theory from Northeastern University, Shenyang, China, in 2004. He is currently an associate professor in Shenyang Ligong University, China. His research interests include data mining, machine learning, especially anomaly detection in online social data. He has published nearly 20 papers in many international/domestic academic journals and conferences.

**Songmin Gu** received his B.S. degree in Computer Science and Technology from Nanjing University of Posts and Telecommunications, China, in 2015. He is currently a master's student in Shenyang Ligong University. His research interests include data mining, machine learning. He has won the secondary prize of Huawei Cup in the 13th domestic master student mathematical modeling contest and the first prize in the first big data contest in Liaoning Province.

**Xiangnan Zhao** received her B.S. degree in Economics from Dalian Nationalities University, China, in 2015. She is currently a master's student in Shenyang Ligong University. Her research interests include data mining and machine learning. She has won the first prize in the first big data contest in Liaoning Province.

**Xiaowei Xu** is a tenured full professor in the Department of Information Science at the University of Arkansas at Little Rock (UALR). He received his Ph.D. in computer science from the University of Munich. Prior to his appointment at UALR, Dr. Xu was a senior research scientist and the head of a research team at Siemens. Dr. Xu is a recipient of the prestigious ACM SIGKDD Test of Time Award for his pioneered work in the density-based clustering algorithm DBSCAN. He is a program committee member and session chair for premier forums including ACM SIGKDD Conference on Knowledge Discovery and Data Mining. He is a consultant for companies and government agencies including Siemens, Acxiom,and FDA. Dr. Xus research focuses on data mining, machine learning, and deep learning. Published over 80 research papers with over 15863 citations, he is one of the most cited researchers according to Google Scholar.