Spotting Collusive Behaviour of Online Fraud Groups in Customer Reviews (Supplementary)

1 Statistical Significance Testing of Hypothesis 1

Hypothesis 1 (Coherence Principle) *Fraudsters in a group are coherent in terms of – (a) the products they review, (b) ratings they give to the products, and (c) the time of reviewing the products.*

To measure the statistical significance of each component (say, products reviewed by the group members), we randomly generate pairs of reviewers (irrespective of the groups they belong to, and form SET 1) and measure how their co-reviewing patterns (cumulative distribution) are different from the case if a pair of reviewers co-occur together in the same group (SET 2). We consider 10,000 such pairs for each group. We hypothesize that both these patterns are different, whereas null hypothesis rejects our hypothesis. We consider three variations for statistical analysis:

1. Rating deviation: Keeping the product same, we select pairs and plot the CDF curves for rating differences between both the groups (Figure 1(a)).
2. Temporal deviation: Keeping the product same we select pairs and plot the CDF curves for time deviations both the groups (Figure 1(b))
3. Jaccard similarity: Distribution of the Jaccard similarity (JS) of the product sets of pairs separately (Figure 1(c))

Rating and temporal deviation of reviewers belonging to the same group (SET 2) are lesser as compared to those in different groups (SET 1) while JS is higher in the former which can be validated through the plots. We observe that the difference is statistically significant for time deviations ($p < 0.01$) which signifies that temporal coherence is more important than rating coherence in detecting potential groups. JS within group is higher as more percentage of groups are having high values.

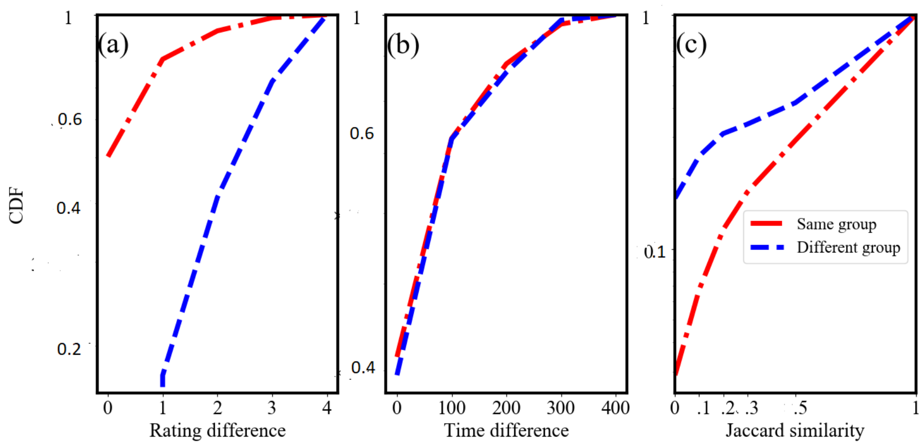

Figure 1: CDF of (a) rating deviation (P value$<$0.02),(b) Time deviation (P value$=$0.005) and (c) Jaccard similarity of product sets (P value$=$0.009) of 10,000 random pairs from both SET 1 (blue lines) and SET 2 (red lines) for YelpNYC.

2 Convergence of `ExtractGroups`

Theorem 2.1 (Theorem of convergence) `ExtractGroups` *will converge within finite iterations.*

Proof Sketch: **Case 1: Isolated nodes.** The algorithm removes isolated nodes at each iteration.
Case 2: Connected components with size greater than 2. The whole component will be removed from the graph.
Case 3: Connected components with size less than or equal to 2. In the next iteration, it will get converted to either of the following three structures:
(i) An isolated node (if the edge does not have a common reviewer with any other edge in the graph). Then Case 1 will follow.
(ii) A connected component with two nodes (if the edge has a common reviewer with only one adjacent edge). Then Case 2 will follow.
(iii) A connected component with more than two nodes (if the edge has a common reviewer with more than one adjacent edges). Then Case 3 will follow.
Lines 15 - 21 do not change the graph and will therefore not come under any of cases stated above. The procedure will eventually lead to a network with no edge.

3 Time Complexity of `ExtractGroups`

Let E be the edge set of the attributed product-product graph. For each iteration after conversion to line graph, there can be at most E isolated nodes or at most $\frac{|E|}{2}$ connected components with 2 nodes leaving the rest as connected components with more than 2 nodes, which will be removed in the current iteration and will not be proceed further.

Algorithm 1 `ExtractGroups`: Candidate Group detection

```
Initialize:
    $CCgroups \leftarrow$ set()                                  ▷ Set of candidate groups
    $G(V, E) \leftarrow$ edge attributed graph
    $CSet \leftarrow \{\}$                                        ▷ Potential merged groups
$G, CCgroups \leftarrow$ GroupDetector($G, CCgroups$)
Iterate:
    $G'(V', E') \leftarrow$ Attributed line graph of G
    $G', CCgroups \leftarrow$ GroupDetector($G', CCgroups$)
    $G \leftarrow G'$
until $|E| > 1$
return $CCgroups$
function GroupDetector($G, CCgroups$)
    for each isolated node $v_i$ in $G$ do
        $CCgroups.add(v_i)$ and remove $v_i$ from $G$
    for each pair of $(e_i, e_j) \in E \times E$ do
        if $a_i^e \subset a_j^e$ then
            if $\frac{\bigcap_{m \in a_j^e} P_m}{\bigcup_{m \in a_j^e} P_m} > 0.5$ then
                $CSet(a_i^e) = CSet(a_i^e) \cup a_j^e$
            else
                if $\frac{\bigcap_{m \in \{a_j^e \setminus a_i^e\}} P_m}{\bigcup_{m \in \{a_j^e \setminus a_i^e\}} P_m} > 0.5$ then
                    $CCgroups.add(a_j^e \setminus a_i^e)$
    for Each $e_i \in E$ do
        $k$=$CSet(a_i^e)$
        $CCgroups.add(k)$ and remove $k$ from $G$
    for Each connected component $c$ with $|c| > 2$ do
        $CCgroups.add(c)$ and remove $c$ from $G$
    for Each group $g \in CCgroups$ do
        if $CollectiveScore(g) <= \tau_{spam}$ then
            $CCgroups.remove(g)$
    return $G, CCgroups$
end function
```

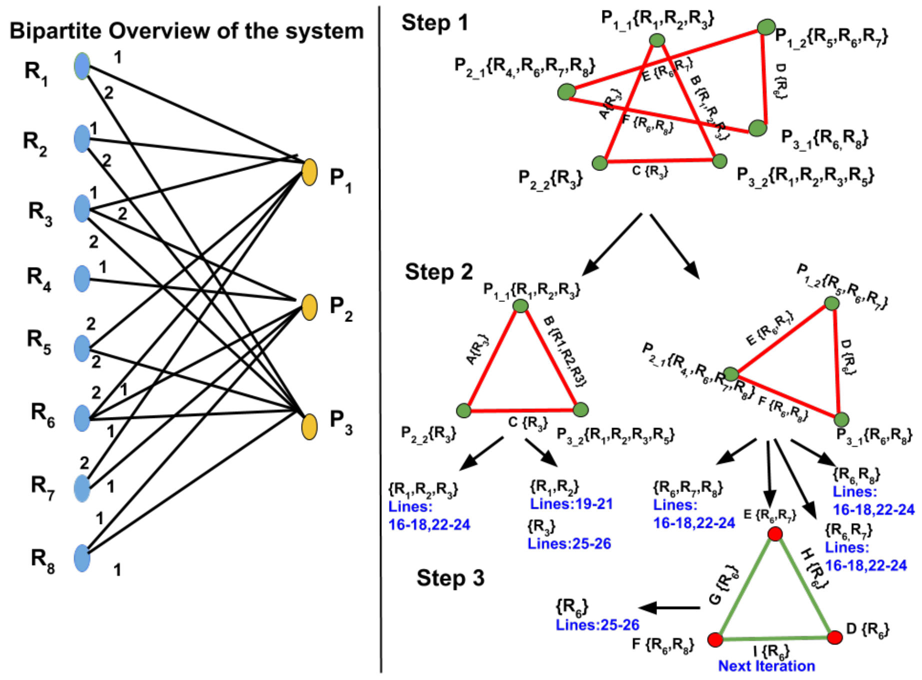

Figure 2: Demonstration of `ExtractGroups`.

Therefore at most E edges will be proceed to the next iteration making complexity of each iteration as less than or equal to $|E|^2$. Lines 16 - 21 will also have worst case complexity as $|E|^2$ (when all edge attributes are distinct). Therefore, the total time complexity is $\mathcal{O}(k|E|^2)$, where k is the number of iterations.

4 Demonstration of `ExtractGroups`

We provide a schematic diagram of the Algorithm 1 in Figure 2. Left panel is a bipartite graph view of the overall system with edges linking reviewers with their reviewed products. We create attributed product-product graph $G(V,E)$ such that each $v_{ij} \in V$ indicates a product-rating pair (p_i, r_j) and its attribute a^v_{ij} consists of the set of reviewers who rated p_i with r_j. An edge $e_{(ij,mn)} = \langle u_{ij}, v_{mn} \rangle \in E$ indicates the co-reviewing and co-rating patterns of two products p_i and p_m with rating r_j and r_n, respectively. The edge attribute $a^e_{(ij,mn)}$ indicates the set of co-reviewers $R_{(ij,mn)}$ who reviewed both p_i and p_m and gave same ratings r_j and r_n respectively at the same time τ_t. Note that edge $e_{(ij,in)}$ connecting same product with different ratings wont exist in G as we assume that a reviewer is not allowed to give multiple reviews/ratings to a single product. Here P_{1_1} has been attributed by $\{R_1, R_2, R_3\}$ as they have given it rating 1 within time τ_t . In a similar fashion we associate a list of users for each node and connect 2 nodes if they have common reviewers between them and store those reviewers as edge attributes. (Step1).

`ExtractGroups` takes this graph as input and executes a series of operations via `GroupDetector`. Isolated nodes are first removed (Lines 13-14), but currently there are none. The edges with Jaccard similarity (JS) of product sets that the corresponding reviewers reviewed greater than a threshold (set as 0.5) are merged. For eg. here R_3 is subset of $\{R_1, R_2, R_3\}$. If JS of product sets of their union, i.e., $\{R_1, R_2, R_3\}$ is greater than 0.5 we will consider $\{R_1, R_2, R_3\}$ as potential group subject to merging with others as the algorithm progresses (Line 16-18). If the union's JS is less than the threshold we report $\{R_1, R_2\}$ who have co reviewed P_1 and P_3 (subject to JS condition) as candidate groups (Line 20-21). All connected components with more than 2 nodes are removed. Here component corresponding to R_3 will be removed. Similarly either $\{R_6, R_7, R_8\}$ will be considered or R_6 if they satisfy the conditions else we move to the next iteration. (Step2)

The remaining structure of G is converted into an attributed line graph (edges converted into vertex and vice versa) $G'(V', E')$ as follows: $v'_{(ij,mn)} \in V'$ corresponds to $e_{(ij,mn)}$ in G and $a^{v'}_{(ij,mn)} = a^{e}_{(ij,mn)}$; an edge $e'_{(ij,mn,ab)} = \langle v'_{(ij,mn)}, v'_{(ij,ab)} \rangle$ represents co-reviewing and co-rating patterns of products p_i, p_m and p_a; the corresponding edge attribute is $a_{(ij,mn,ab)} = a^{e}_{(ij,mn)} \cap a^{e}_{(ij,ab)}$ (Figure). G' is again fed into `GroupDetector` in the next iteration. Again no isolated nodes are there. Edges D, E and F all have R_6 in common, so they will form a connected component with more than 2 nodes and will be reported as a candidate group, It will be removed from the graph thus removing all the edges (termination condition, Line 10). (Step3)

We define $CollectiveScore$ as the average of six group level indicators and consider those as potential groups whose $CollectiveScore$ exceeds τ_{spam}. Let us assume that $\{R_1, R_2\}$, R_3 and R_6 are the groups that satisfied the given conditions. So $\{R_1, R_2\}$ (after removal of groups with size less than 2) will be reported as the potential candidate group on termination.