Deal or Deceit: Detecting Cheating in Distribution Channels

Kai Shu[1,2], Ping Luo[1], Wan Li[2], Peifeng Yin[3], Linpeng Tang[4]
[1]Key Lab of Intelligent Information Processing of Chinese Academy of Sciences (CAS), Institute of Computing Technology, CAS, Beijing 100190, luop@ict.ac.cn
[2]Chongqing University, Chongqing 400044, China, {kai.shu, wanli}@cqu.edu.cn
[3]Pennsylvania State University, pzy102@cse.psu.edu
[4]Princeton University, linpengt@cs.princeton.edu

ABSTRACT

Distribution channel is a system that partners move products from manufacturer to end users. To increase sales, it is quite common for manufacturers to adjust the product prices to partners according to the product volume per deal. However, the price adjustment is like a double-edged sword. It also spurs some partners to form a cheating alliance, where a *cheating seller* applies for a *falsified* big deal with a low price and then re-sells the products to the *cheating buyers*. Since these cheating behaviors are harmful to a healthy ecosystem of distribution channel, we need the automatic method to guide the tedious audit process.

Thus, in this study we propose the method to rank all partners by the degree of cheating, either as seller or buyer. It is mainly motivated by the observation that the sales volumes of a cheating seller and its corresponding cheating buyer are often negatively correlated with each other. Specifically, the proposed framework consists of three parts: 1) an asymmetric correlation measure which is needed to distinguish cheating sellers from cheating buyers; 2) a systematic approach which is needed to remove false positive pairs, i.e., two partners whose sale correlation is purely coincident; 3) finally, a probabilistic model to measure the degree of cheating behaviors for each partner.

Based on the 4-year channel data of an IT company we empirically show how the proposed method outperforms the other baseline ones. It is worth mentioning that with the proposed *unsupervised* method more than half of the partners in the resultant top-30 ranking list are true cheating partners.

Categories and Subject Descriptors

H.2.8 [**Database Applications**]: Data mining

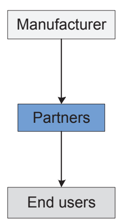
Keywords

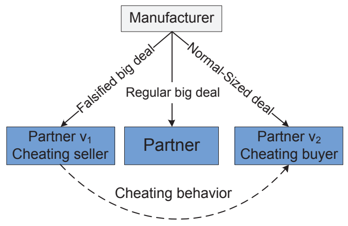
Distribution Channel; time series; correlations

1. INTRODUCTION

Distribution channel, as shown in Figure 1a, is a system of *partners* to move products from *manufacturer* to *end users*. When the number of end customers is large, this indirect channel is extremely helpful to improve product revenue and market competitiveness. To maintain the smooth operation of distribution channel, manufacturers need to develop effective *channel policies*, addressing the following aspects: (i) ensure that end users can purchase products at any time and place, (ii) control product price reasonably, (iii) increase market promotion and market share. Among those goals, price management may be the most important and critical one. To spur the sales enthusiasm of channel partners, manufacturer may adjust the product prices according to sales volume per deal. If a partner has an end user who plans to buy plenty of products, she can apply for a low price from the manufacturer. On the contrary, for a normal-sized deal the partner can only get a normal price. Thus, price may differ among partners.

CIKM'14, November 03–07, 2014, Shanghai, China.
Copyright 2014 ACM 978-1-4503-2598-1/14/11 ...$15.00.
http://dx.doi.org/10.1145/2661829.2661874 .





(a) Indirect channel **(b) Cheating behavior**
Figure 1: The cheating behavior in distribution channel

Due to such price differences, *cheating* in the distribution channel may happen. Hardy and Magrath [6] summarized the types and forms of channel cheating, and then explored the structures, factors and incentives that encourage cheating. Additionally, Narayandas and Rangan [16] showed that to achieve relational benefits and competitive advantages, partners have put emphasis on developing a stable dyadic relationship instead of building adversarial relationships with other partners. Thus, cheating becomes the behaviors of the partners' alliance nowadays.

A typical scenario of *cheating alliance* is shown in Figure 1b. Partner v_1 pretends to have an end user with a falsified big deal, thus applies for a low price from the manufacturer. Then, v_1 re-sells part of products to another partner v_2 with the price lower than the regular price of the normal deals. In this scenario, we call v_1 and v_2 *cheating seller* and *cheating buyer* respectively. However, from the view point of the manufacturer such behaviors, represented by the dash arrows in Figure 1b, are totally *unknown.* Only the sales volume from the manufacturer to the partners, represented by the solid arrows, are *known.* To cultivate a healthy ecosystem of distribution channel, these unknown cheating behaviors among partners must be detected.

Usually, the audit staff in a company is responsible to detect these cheating behaviors, and their work heavily relies on the process of official examination on the business and financial records. For some severe cheating cases the judicial investigation is also required. It is really a tedious process. Therefore, in this study we aim to provide an automatic method to guide the audit process and greatly reduce the manual efforts needed.

Motivating observation. The only data we can use are the purchase volumes of the partners from the manufacturer. Here, the purchase volume of each partner can be represented as a time series, in which each entry corresponds to the monthly purchase volume. Figure 2 illustrates the time series of purchase volume of two real-world cheating partners, where v_1 and v_2 are the paired cheating seller and cheating buyer, respectively. Usually, the sales volume of these two partners may change collectively on the following two aspects.

• When v_1 applies for a falsified big deal from the manufacturer, its purchase volume increases at that month. Meanwhile, since v_2 buys some products from v_1, its purchase volume from the manufacturer will decrease after v_1's big deal happens. For example, in Figure 2 v_1's purchase volume increases in the 5th month while v_2's purchase volume decreases in the 6th month.

• On the contrary, if v_1 does not apply for big deals, its purchase volume decreases. At this time, as v_2 can only purchase the products from the manufacturer, its purchase volume will increase (if she has the stable product requirements). For example, v_1's purchase volume decreases in the 7th month while v_2's purchase volume increases on the 7th month.

Intuitively, such phenomenon may to some degree be treated as evidence for cheating behavior. The higher the frequency of the phenomenon is, the stronger the evidence would be.

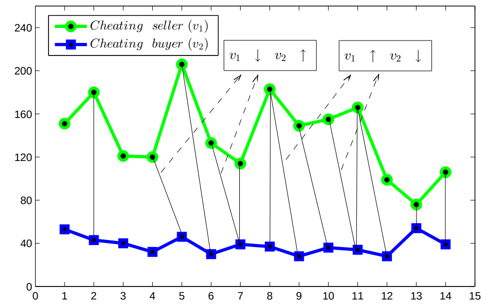


Figure 2: The time series of purchase quantities of two real-world partners with cheating behaviors

Motivated by the above observations, in this study we propose the method to detect the cheating behaviors among the partners. To the best of our knowledge, compared with some previous qualitative works this is the first quantitative study in this area. In this problem, we are given the sequences of purchase volume for all the partners, and aim to rank the partners based on the degree of their cheating behaviors. This ranking list can be delivered to the audit staff for further investigation sequentially.

In the following we will present the framework of our method, discuss the challenges in each step of this framework, and highlight the technical contributions.

2. DETECTION FRAMEWORK

The framework of the solution is shown in Figure 3. It mainly consists of three steps:

1) Building the *Partner Correlation Graph*;
2) Partition partners into suspicious sellers and buyers;
3) Ranking the partners by the probabilistic model.

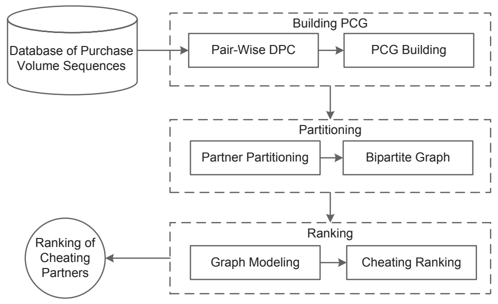


Figure 3: The framework of cheating detection

2.1 Building the Partner Correlation Graph

Motivated by the observation shown in Figure 2, we actually need a method to measure the correlation between two sequences. In this study, we adapt the *Pearson correlation* to depict the collective changes of the purchase volumes between a cheating seller and a cheating buyer. Intuitively, the opposite tendencies of the two sequences yield a strong *negative* correlation between them. Thus, we believe that if two sequences have strong negative correlation, it is likely that the abnormal deals happen between the corresponding partners.

However, Pearson correlation has the following limitations in this application. First, Pearson correlation is a symmetric measure, and the order of its two inputs is irrelevant to its output. It cannot distinguish cheating sellers from cheating buyers in this application. Thus, an asymmetric measure is needed here. Second, Pearson correlation is computed with the tick-to-tick correspondence on the two sequences. However, an abnormal deal does not necessarily complete during the same month, but with a delay of several months after the cheating seller applies for a big deal from the manufacturer. Thus, we need to consider this time offset in computing the correlation.

To address these challenges, we leverage the *Dynamic Time Warping* (DTW) technique to compute a *Directed Pearson Correlation* (DPC) coefficient. DTW is a well-known method to find an optimal time alignment between two sequences under certain constraints. In an abnormal deal, the cheating seller can only sell products to the cheating buyer in or near after the time when the cheating seller gets a falsified big deal. Thus, the warping direction must be *from now to future* within a time window.

To this end, we propose an asymmetric measure r_{dpc} for the two sequences $\vec{x}_1, \vec{x}_2$ of the two partners v_1, v_2. $r_{dpc}(\vec{x}_1, \vec{x}_2)$ measures the correlation when we view v_1 as a cheating seller and v_2 as a cheating buyer. Similarly, $r_{dpc}(\vec{x}_2, \vec{x}_1)$ measures the correlation when we view v_2 as a cheating seller and v_1 as a cheating buyer. Note that $r_{dpc}(\vec{x}_1, \vec{x}_2) \neq r_{dpc}(\vec{x}_2, \vec{x}_1)$. The details on computing r_{dpc} will be presented in Section 3.

Therefore, given the n partners with their sequences of purchase volumes $\{\vec{x}_i | i = 1, \cdots, n\}$ ($\vec{x}_i$ is the sequence of the partner v_i), we can generate a weighted directed graph $G = (V, E, w)$, where

• $V = \{v_1, \cdots, v_n\}$ contains the n nodes, where v_i corresponds to the partner with the sequence $\vec{x}_i$;

• $E = \{(v_i, v_j) | -r_{dpc}(\vec{x}_i, \vec{x}_j) > \eta\}$, where $0 < \eta < 1$ is a user-specified parameter;

• The weight w_{ij} on the edge of (v_i, v_j) is set to $-r_{dpc}(\vec{x}_i, \vec{x}_j)$.

We call this graph *Partner Correlation Graph* (PCG for short). Remember that we aim to identify the negative correlations among partners. Thus, the weight w_{ij} is set to the *opposite value* of $r_{dpc}(\vec{x}_i, \vec{x}_j)$, and only the directed edges with the weight values bigger than η exist in this graph. Figure 4 gives an example of the generated PCG graph with 5 partners when $\eta = 0.1$. We use this graph as a running example in this paper.

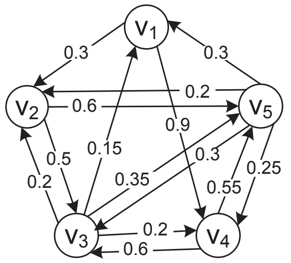


Figure 4: The directed graph of partner correlation

2.2 Partition of Partners

In building the PCG graph we compute the DPC values in both the two directions between each pair of partners. In other words, we view each partner as both cheating seller and cheating buyer. For example, when we consider the edge from v_5 to v_1 in Figure 4, v_1 acts as a cheating buyer. Meanwhile, on the edge from v_1 to v_4, v_1 acts as a cheating seller. However, in real world only the partners with abundant capital can be cheating sellers since a large amount of investment is needed for big deals. Also, these partners have no incentive to be cheating buyers. On the other hand, the partners with less capital can only be cheating buyers. Therefore, the two roles are exclusive, indicating that each partner can have only one character, i.e. cheating seller or cheating buyer[1].

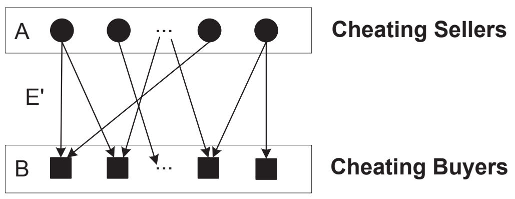


Figure 5: The bipartite graph of candidate cheating partners

To meet this requirement, we need a method to partition the nodes of the graph into two disjoint subsets, i.e. A and B, representing the sets of suspicious cheating sellers and buyers respectively. Also, only the edges going from the source-side A to the sink-side B remain in the graph. Figure 5 gives an example of the resultant bipartite.

To this end, we actually need a *cutting method* for a weighted directed graph $G = (V, E, w)$, aiming to maximize certain objective function of the partition (A, B) of V. In this study, we formulate a new *cut problem* (called *Max-DifCut*), which outputs the partition (A, B) $(A \cap B = \emptyset, A \cup B = V)$ such that

$$3w_{AB} - w_{BA}$$

is maximized, where $w_{AB} = \sum_{i \in A, j \in B} w_{ij}$ is the weight sum of the edges from A to B, and w_{BA} is defined similarly.

Section 4 will detail why we formulate this new problem based on the application background. As it is a NP-hard problem, we give the greedy algorithm for it. The experimental results will empirically show that the ranking performance will be significantly improved if the ranking is based on the resultant bipartite graph after removing the noisy edges.

2.3 Ranking Partners by Generative Probabilistic Model

Next, we propose the generative probabilistic model to fit the resultant bipartite with the edge weights. In this probabilistic model, we assume each partner has two latent variables α and β, representing the cheating degree as a cheating seller and a cheating buyer respectively. Specifically, bigger α indicates higher motivation of being a cheating seller and smaller β suggests higher motivation of being a cheating buyer. We suppose that the *cheating probability* that partner v_1 sells products to v_2 can be modeled as a Beta function with the parameters of the α value of the source node and the β value of the sink node. The modeling process is to fit the model to the graph so that the cheating probability that any partner sells products to another approximates as much as possible to the weight on the corresponding edge.

With the model parameters we develop the ranking score for each partner, measuring the degree of cheating either as seller or buyer. All these details will be presented in Section 5.

3. DIRECTED PEARSON CORRELATION

In this section, we detail the computing of DPC to build the PCG graph. First, we introduce the preliminaries on Pearson correlation and dynamic time warping. Then, we show how to compute DPC with the DTW technique. Here, let $\vec{x} = (\vec{x}(1), \vec{x}(2), ..., \vec{x}(m))$ and $\vec{y} = (\vec{y}(1), \vec{y}(2), ..., \vec{y}(m))$ be the two sequences of purchase volumes from the two partners in the past m months, respectively.

3.1 Pearson Correlation

In statistics, a measure of correlation is a numerical value which describes the strength of a relationship among variables. Pearson correlation is widely used for analyzing numerical variables such as time series. Giving two sequences $\vec{x}$ and $\vec{y}$, Pearson correlation can be computed in Equation (1),

$$r(\vec{x}, \vec{y}) := \frac{1}{m-1} \sum_{t=1}^{m} \left(\frac{\vec{x}(t) - \overline{x}}{\delta_{\vec{x}}}\right)\left(\frac{\vec{y}(t) - \overline{y}}{\delta_{\vec{y}}}\right) \tag{1}$$

where $\overline{x}$ $(\overline{y})$ denotes the average value of the entries in $\vec{x}$ $(\vec{y})$ and $\delta_{\vec{x}}$ $(\delta_{\vec{y}})$ denotes the standard deviation of $\vec{x}$ $(\vec{y})$.

Note that the correlation is computed via the tick-to-tick correspondence by multiplying $\frac{\vec{x}(t)-\overline{x}}{\delta_{\vec{x}}}$ and $\frac{\vec{y}(t)-\overline{y}}{\delta_{\vec{y}}}$ at the same time stamp t. Moreover, it is symmetric since $r_{(\vec{x},\vec{y})} = r_{(\vec{y},\vec{x})}$ holds. Pearson correlation ranges from -1 to 1. $r = 1$ implies that a linear equation describes the relationship between $\vec{x}$ and $\vec{y}$ perfectly, with all data points lying on a line for which $\vec{y}$ increases as $\vec{x}$ increases. $r = -1$ implies that all data points lying on a line for which $\vec{y}$ decreases as $\vec{x}$ increases. $r = 0$ implies that there is no linear correlation between them.

However, Pearson correlation has the following limitations in this application. First, Pearson correlation is a symmetric measure, and the order of its two inputs is irrelevant to its output. It cannot distinguish cheating sellers from cheating buyers in this application. Thus, an asymmetric measure is needed in this application. Second, Pearson correlation is computed with the tick-to-tick correspondence on the two sequences. However, an abnormal deal does not necessarily complete during the same month, but with a delay of several months after the cheating seller applies a big deal from the manufacturer. Thus, we need to consider this time offset in computing the correlation.

3.2 Dynamic Time Warping

Next, we briefly introduce the technique of dynamic time warping, which can be used to transform Pearson correlation into an asymmetric measure. Dynamic time warping [20] is a transformation that allows sequences to be stretched along the time axis to minimize the distance between them. Giving two sequences $\vec{x} = (\vec{x}(1), \vec{x}(2), ..., \vec{x}(m))$ and $\vec{y} = (\vec{y}(1), \vec{y}(2), ..., \vec{y}(m))$, a *local cost measure* $c(\vec{x}(i), \vec{y}(j))$ is denoted to represent the distance between $(\vec{x}(i), \vec{y}(j))$. Thus, we can obtain a *cost matrix* $C \in \mathbb{R}^{m \times m}$ defined by $C(i, j) = c(\vec{x}(i), \vec{y}(j))$. Here, an alignment is a warping path $p =$

[1]To simply this problem, we also assume that the character of a partner does not change along the time.

$(p_1, ..., p_L)$ with $p_l = (i_l, j_l) \in [1, ..., m] \times [1, ..., m]$ for $l \in [1, ..., L]$ satisfying the following three conditions,

- Boundary condition: $p_1 = (1, 1)$ and $p_L = (m, m)$.
- Monotonicity condition: $i_1 \le i_2 \le ... \le i_L$ and $j_1 \le j_2 \le ... \le j_L$.
- Step size condition: $p_{l+1} - p_l \in \{(1, 0), (0, 1), (1, 1)\}$ for $l \in [1 : L - 1]$.

The total cost of a warping path p is defined in Equation (2),

$$c_p(\vec{x}, \vec{y}) := \sum_{l=1}^{L} c(\vec{x}(i_l), \vec{y}(j_l)) \tag{2}$$

Here, L is the length of the warping path.

An *optimal warping path* between $\vec{x}$ and $\vec{y}$ is a warping path that have minimal total cost. The DTW distance $D(\vec{x}, \vec{y})$ of two sequences $\vec{x}$ and $\vec{y}$ is the sum of tick-to-tick distances after the two sequences have been optimally warped to match each other, which is denoted in Equation (3),

$$d(\vec{x}, \vec{y}) := \min\{c_p(\vec{x}, \vec{y}) \mid p \text{ is a warping path}\} \tag{3}$$

To determine the optimal warping path, a dynamic programming algorithm was proposed to solve it. Let $d(i, j)$ denotes the accumulate distance of (i, j), then the iterative condition holds as in Equation (4),

$$\begin{aligned} d(i, j) := c(\vec{x}(i), \vec{y}(j)) + \min\{&d(i - 1, j - 1), \\ &d(i, j - 1), d(i - 1, j)\} \end{aligned} \tag{4}$$

3.3 Directed Pearson Correlation

As discussed in Sections 2.1 and 3.1 we need an asymmetric correlation measure which also considers the time offset from the source sequence to the sink sequence. Thus, we adapt the original Pearson correlation by using the DTW technique. First, we define the following *local cost measure* as in Equation (5),

$$c'(\vec{x}(i), \vec{y}(j)) := \frac{\vec{x}(i) - \overline{x}}{\delta_{\vec{x}}} \cdot \frac{\vec{y}(j) - \overline{y}}{\delta_{\vec{y}}} \tag{5}$$

Similarly, we can define the total cost as Equation 6,

$$c'_p(\vec{x}, \vec{y}) := \sum_{l=1}^{L} c'(\vec{x}(i_l), \vec{y}(j_l)) = \sum_{l=1}^{L} (\frac{\vec{x}(i_l) - \overline{x}}{\delta_{\vec{x}}})(\frac{\vec{y}(j_l) - \overline{y}}{\delta_{\vec{y}}}), \tag{6}$$

where $(i_l, j_l) = p_l$ denotes an element of a warping path p. Then, we can find an optimal alignment $p^* = (p_1^*, ..., p_L^*)$, which has the minimal total cost $c'_{p^*}(\vec{x}, \vec{y})$.

To avoid degenerated matching, by which we mean a relatively small section of one sequence maps onto a relatively large section of another, the warping path is often limited by global constraints. The warping scope s is the area that the warping path is allowed to visit in warping matrix. The Sakoe-Chiba band [18] is a well-known constraint that restricts the warping path to the range of $|i_l - j_l| \le s$. However, in our application a cheating seller can only sell the products to a cheating buyer in or near after the month. To address this issue, we propose a more restricted constraint, namely $0 \le j_l - i_l \le s$. As illustrated in Figure 6, when $s = 2$ the black and grey areas denote all probable warping elements between $\vec{x}$ and $\vec{y}$ under this constraint, and the black areas denote the optimal warping path.

Thus, the DPC value of two sequences $\vec{x}$ and $\vec{y}$, denoted by $r_{dpc}(\vec{x}, \vec{y})$, is given in Equation (7),

$$\begin{aligned} r_{dpc}(\vec{x}, \vec{y}) := \min\{\frac{1}{L-1} c'_p(\vec{x}, \vec{y}) \mid &p \text{ is a warping path} \\ &\text{satisfying the constraint, } L = |p|\} \end{aligned} \tag{7}$$

Note that it is easy to prove that $r_{dpc}(\vec{x}, \vec{y}) \le r(\vec{x}, \vec{y})$ always holds for any two sequences $\vec{x}$ and $\vec{y}$ since Pearson correlation is a special case of DPC when $s = 0$. Also, it is clear that $r_{dpc}(\vec{x}, \vec{y}) \ne r_{dpc}(\vec{y}, \vec{x})$, indicating $r_{dpc}(\vec{x}, \vec{y})$ is an asymmetric measure.

Algorithm 1 gives the process of computing r_{dpc}. After the initialization in Lines 1 through 8, Lines 9 through 14 compute the optimal warping value. Line 15 gets the size of optimal warping path. Line 16 computes the value of r_{dpc}.

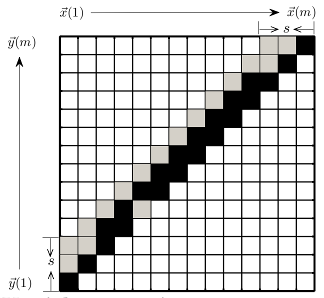


Figure 6: DTW with a warping scope constraint s $(s = 2)$

Algorithm 1 Computing DPC

```
Input:
    two sequences: $\vec{x}$ and $\vec{y}$
    the user-specified warping scope $s$
Output:
    $r_{dpc}(\vec{x}, \vec{y})$
 1: $m := |x|$
 2: $dtw[] := new[m \times m]$
 3: for $i := 0$; $i < m$; $i++$ do
 4:     for $j := 0$; $j < m$; $j++$ do
 5:         $d(i, j) := \infty$
 6:     end for
 7: end for
 8: $d(0, 0) := 0$
 9: for $i := 1$; $i < m$; $i++$ do
10:     for $j := i$; $j < m$ and $j \le i + s$; $j++$ do
11:         $c := \frac{\vec{x}(i) - \overline{x}}{\delta_{\vec{x}}} \cdot \frac{\vec{y}(j) - \overline{y}}{\delta_{\vec{y}}}$
12:         $dtw(i, j) := c + \min\{d(i - 1, j),$
            $d(i, j - 1), d(i - 1, j - 1)\}$
13:     end for
14: end for
15: $L := |p^*|$
    //the length of the optimal warping path $p^*$
16: $r_{dpc}(\vec{x}, \vec{y}) := \frac{1}{L-1} \times dtw(m - 1, m - 1)$
17: return $r_{dpc}(\vec{x}, \vec{y})$
```

With the proposed DPC measure we can exhaustively compute this value on the two directions of each pair of partners, and then build the PCG graph as discussed in Section 2.1. Figure 4 shows an example of this graph. Note that only the edges with strong negative correlations remain in the resultant PCG graph.

4. PARTITION OF PARTNERS

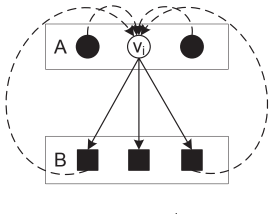
In the PCG graph it is possible that a node has both out-edges and in-edges. The nodes with both out-edges and in-edges are considered as both cheating sellers and cheating buyers simultaneously. However, in reality each partner can act as only one character, namely cheating seller or cheating buyer. With this requirement we need a method to partition the nodes of the graph into two disjoint subsets, i.e. A and B, representing the sets of suspicious cheating sellers and buyers respectively. Only the edges going from the source-side A to the sink-side B remain in the graph. In other words, this method is to remove some edges, ensuring that each partner can only have either in-edges or out-edges.

Specifically, the task is to find a *cut set* in a directed weighted graph such that certain objective function of the partition (A, B) of V is maximized. It can be used to remove the noisy edges in the PCG graph. The empirical experiments will show that the ranking performance based on the resultant bipartite is significantly better than that on the PCG graph. All the notations used here are summarized in Table 1.

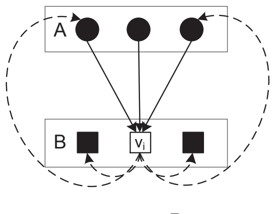
Table 1: List of Symbols

Symbol	Meaning
A	the node set for cheating sellers
B	the node set for cheating buyers
w_{AB}	the weight sum of all the edges from A to B
w_{iB}	the weight sum of all the edges from node v_i to all the nodes in B
w_{Aj}	the weight sum of all the edges from all the nodes in A to node v_j
w_{i*}	the weight sum of all the edges from node v_i to all the other nodes
w_{*j}	the weight sum of all the edges from all the other nodes to node v_j
w_G	the total weight of edges in G

4.1 Problem Formulation of Max-DifCut



(a) Δw_i^A



(b) Δw_i^B

Figure 7: The illustration of cheating degree for node v_i in the bipartite

Most of the previous studies in this area focus on the problem of *Max-DiCut* [14], which outputs the partition (A, B) of $V(A \cap B = \emptyset, A \cup B = V)$ such that w_{AB} is maximized, where $w_{AB} = \sum_{i \in A, j \in B} w_{ij}$ is the weight sum of the edges from A to B.

In this work, we formulate a new graph cut problem, called *Max-Difference CUT* (*Max-DifCut* for short), on the directed weighted graph. In the following we detail how this new problem is motivated by the application background of cheating detection.

Assume we have a partition (A, B) of V, where A and B are the sets of cheating sellers and buyers respectively. Based on this partition, we can propose the following measure Δw_i to check the degree that node v_i is a cheating partner, either as seller or buyer,

$$\Delta w_i = \begin{cases} \Delta w_i^A = w_{iB} - w_{*i}, & \text{if } v_i \in A \\ \Delta w_i^B = w_{Ai} - w_{i*}, & \text{if } v_i \in B \end{cases} \tag{8}$$

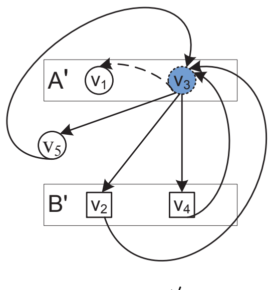
In the above equation we consider the two situations, namely $v_i \in A$ (when v_i is a cheating seller) and $v_i \in B$ (when v_i is a cheating buyer). First, as shown in Figure 7(a), when $v_i \in A$ this measure is defined as $w_{iB} - w_{*i}$. Here, w_{iB} is the weight sum of the edges from node v_i to any node in B (the weight sum of all the solid-arrow lines), actually measuring the amount of all the possible cheating behaviors if node v_i is a cheating seller; w_{*i} is the weight sum of the edges from any other node to node v_i (the weight sum of all the dash-arrow lines), measuring the amount of noises if node v_i is a cheating seller. Thus, the bigger this difference of $w_{iB} - w_{*i}$, the more likely that node v_i is a cheating seller. Second, as shown in Figure 7(b), when $v_i \in B$ this measure is defined as $w_{Ai} - w_{i*}$. Similarly, the weight sum of solid-arrow lines represents w_{Ak} and that of dash-arrow lines represent w_{i*}. The bigger this difference of $w_{Ai} - w_{i*}$, the more likely that node v_i is a cheating buyer.

Therefore, we aim to find the partition in which this measure on each node is as big as possible. By adding these measures on all the nodes we get the objective function as in Formula 9 and can transform it into Formula 10,

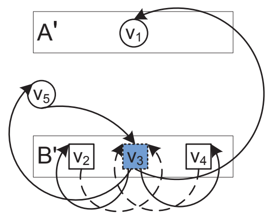
$$\sum_{v_i \in A} (w_{iB} - w_{*i}) + \sum_{v_j \in B} (w_{Aj} - w_{j*}) \tag{9}$$
$$= (w_{AB} - w_{*A}) + (w_{AB} - w_{B*})$$
$$= 3w_{AB} - w_{BA} - w_G \tag{10}$$

Note that w_G in Formula 10 is a constant for any graph G. Therefore, we aim to find a partition (A, B) of E such that $3w_{AB} - w_{BA}$ is maximized.

4.2 Solution to Max-DifCut



(a) $\Delta w_3^{A'}$



(b) $\Delta w_3^{B'}$

Figure 8: The illustration for $\Delta w_3'$ in Max-DifCut. Only the edges on v_3 are shown

Similar to the classical *Max-DiCut* problem, the hardness of *Max-DifCut* is also NP-hard [2]. Thus, we need the approximate algorithm to solve it. In this study we develop a greedy algorithm for this problem [2].

As shown in Algorithm 2, it performs iteratively. Let A', B' be the sets of cheating sellers and cheating buyers respectively in the current round. At the beginning, A', B' are both empty sets. Then, in each round we select one node which maximizes certain greedy function, and put it into A or B. Based on the problem formulation, Equation (8) can be naturally adapted as the greedy function. Specifically, this greedy function $\Delta w_i'$ is defined as

$$\Delta w_i' = \begin{cases} \Delta w_i^{A'} = (w_{i*} - w_{iA'}) - w_{*i}, & \text{if } v_i \in A' \\ \Delta w_i^{B'} = (w_{*i} - w_{B'i}) - w_{i*}, & \text{if } v_i \in B' \end{cases} \tag{11}$$

Here, w_{iB} and w_{Ai} in Equation (8) are substituted with $(w_{i*} - w_{iA'})$ and $(w_{*i} - w_{B'i})$ respectively. It is based on the fact that for the current sets of A', B', $(w_{i*} - w_{iA'})$ and $(w_{*i} - w_{B'i})$ are actually the *upper bounds* of w_{iB} and w_{Ai} respectively. Take the upper bound of w_{iB} as an example. For v_i, the nodes on the out-edges of v_i is denoted by $V_i^{out} = \{v_j | (i, j) \in E\}$. It can be divided into two parts with the current A', namely $V_i^{out} \cap A'$ and $V_i^{out} - A'$. If we put all the nodes in $V_i^{out} - A'$ into B, then w_{iB} is equal to $(w_{i*} - w_{iA'})$. Thus, for the current A', $(w_{i*} - w_{iA'})$ is the upper bound of w_{iB}.

Then, with the definition of $\Delta w_i'$, Lines 3 through 7 in Algorithm 2 compute $\Delta w_i'$ of all the nodes, and Line 8 chooses the node that maximizes $\Delta w_i'$. We use the example in Figure 4 to describe the running process.

- Round 1. At the beginning with $A' = \{\}, B' = \{\}$, $\Delta w_1^{A'}$ reaches the maximal value. Thus, we put node v_1 into A'. Meanwhile, in order to make Δw_1 in the final partition equal to $\Delta w_1^{A'}$, we also put all the nodes on the out-edges of v_1, namely nodes v_2 and v_4, into B'. This is conducted by Lines 9 through 16 in Algorithm 2. After this round, $A' = \{v_1\}, B' = \{v_2, v_4\}$.
- Round 2. Next, for the second round we compute the values of $\Delta w_3^{A'}, \Delta w_3^{B'}, \Delta w_5^{A'}, \Delta w_5^{B'}$. The sub-figures in Figure 8 illustrate the process for computing $\Delta w_3^{A'}, \Delta w_3^{B'}$ respectively.

As shown in Figure 8(a), we compute $\Delta w_3^{A'}$ if v_3 is put into A'. Here, v_3 has 4 outedges. However, since currently $v_1 \in A'$ the weight on the edge of (v_3, v_1) cannot be considered. Thus, we have

$$\Delta w_3^{A'} = (w_{3*} - w_{31}) - w_{*3} = (w_{35} + w_{32} + w_{34}) - w_{*3} = -0.65$$

[2]The technique of Semi-Definite Programming (SDP) can also be used here, and will be discussed in Section 7 for the related work.

Similarly, as shown in Figure 8(b), we compute $\Delta w_3^{B'}$ if v_3 is put into B'. Here, v_3 has 3 inedges. However, since currently $v_2 \in B', v_4 \in B'$ the weights on the edges of (v_2, v_3) and (v_4, v_3) cannot be considered. Thus, we have

$$\Delta w_3^{B'} = (w_{*3} - (w_{23} + w_{43})) - w_{3*} = w_{53} - w_{3*} = -0.6$$

Again, we can get $\Delta w_5^{A'} = -0.75$, $\Delta w_5^{B'} = -0.7$. Clearly, $\Delta w_3^{B'}$ is the maximal value among $\Delta w_3^{A'}, \Delta w_3^{B'}, \Delta w_5^{A'}, \Delta w_5^{B'}$. Thus, we put node v_3 into B'. Then, we consider all the nodes on the in-edges of node v_3, namely v_2,v_4,v_5. Since v_2 and v_4 are already in B', we can only put v_5 into A'. Finally, the partition result is as follows,

$$A = \{v_1, v_5\}, \qquad B = \{v_2, v_3, v_4\}$$

Algorithm 2 The Greedy Algorithm (Max-DifCut)

```
Input:
    a directed graph $G = (V, E, w)$
Output:
    a bipartite graph with $E' := \{(v_i, v_j) \in E$
    $| \; v_i \in A$ and $v_j \in B\}$
 1: $n := |V|$, $A', B' := \{\}$
    /* $A'$ and $B'$ denote current set of A and B */
 2: while $A' \cup B' \neq V$ do
 3:   for all $i \notin A'$ and $i \notin B'$ do
 4:     $\Delta w_i^{A'} := (w_{i*} - w_{iA'}) - w_{*i}$
 5:     $\Delta w_i^{B'} := (w_{*i} - w_{B'i}) - w_{i*}$
 6:     $\Delta w_i' := \max(\Delta w_i^{A'}, \Delta w_i^{B'})$
 7:   end for
 8:   $k := \arg\max_i \Delta w_i'$
 9:   if $\Delta w_k' = \Delta w_k^{A'}$ then
10:     add $v_k$ to $A'$
11:     add all $v_j$ to $B'$ if $(v_k, v_j) \in E_{k*}$ and $v_j \notin A'$
        /* $E_{k*}$ denotes all out-edges of node $v_k$ */
12:   end if
13:   if $\Delta w_k' = \Delta w_k^{B'}$ then
14:     add $v_k$ to $B'$
15:     add all $v_j$ to $A'$ if $(v_j, v_k) \in E_{*k}$ and $v_j \notin B'$
        /* $E_{*k}$ denotes all in-edges of node $v_k$ */
16:   end if
17: end while
18: return $A', B'$
```

5. RANKING PARTNERS

In this section, we will detail the method of ranking partners. We first introduce the basic ranking method which is motivated by partition process. Then we demonstrate a probabilistic ranking model, and give the ranking function based on it.

5.1 Ranking based on Edge Weights of the Bipartite

Here, we discuss the ranking methods using only the edge weights on the resultant bipartite.

Given the partition (A, B) of V generated by the proposed graph cut method, Δw_i in Equation (8) can be naturally adopted as the ranking score, namely

$$score(i) = \Delta w_i \tag{12}$$

we call this ranking method *Cut_Rank*, since the graph cut method is performed to generate the bipartite firstly.

5.2 Probabilistic Model to Generate Partner Correlation Graph

Next, we propose a probabilistic model to generate the weighted graph $G = (V, E, w)$, and use the resultant model parameters to rank partners. Note that this probabilistic model can be applied to any weighted graph G, e.g. the original PCG graph and the bipartite graph after partitioning.

The modeling process is to fit the model to the graph data so that certain objective function is minimized. Here, the objective function consists of two parts, i.e., data fitting and model complexity. The former one guarantees that the trained model represents the observed data while the second one avoids the over-fitting situation.

Formally, let $\boldsymbol{\theta}$ denotes the set of model parameters, $p(w_{ij}|\boldsymbol{\theta})$ denotes the probability that w_{ij} is sampled given $\boldsymbol{\theta}$, and $\mathtt{C}(\boldsymbol{\theta})$ denotes the model complexity function. Then, the objective function $\mathcal{L}(G, \boldsymbol{\theta})$ is defined as

$$\mathcal{L}(G, \boldsymbol{\theta}) = -\sum_{(v_i,v_j)\in E} \log p(w_{ij}|\boldsymbol{\theta}) + \lambda \cdot \mathtt{C}(\boldsymbol{\theta}) \tag{13}$$

where $\lambda \in [0, +\infty]$ is a trade-off parameter to adjust the relative importance between the two terms. In the following we give the design of $p(w_{ij}|\boldsymbol{\theta})$ and $\mathtt{C}(\boldsymbol{\theta})$.

The design of $p(w_{ij}|\boldsymbol{\theta})$. In this model we consider the motivation of being a cheating seller and a cheating buyer, separately. Specifically, we assume that each node v_i has two latent variables, α_i and β_i. The former indicates how much likely the partner will cheat as cheating seller while the latter as cheating buyer. Specifically, bigger α indicates higher motivation of being a cheating seller and smaller β suggests higher motivation of being a cheating buyer.

Given two nodes v_i and v_j, the probability of v_i selling products to v_j, denoted by p_{ij}, is modeled as a Beta distribution in Equation (14).

$$\mathtt{Beta}(p_{ij}; \alpha_i, \beta_j) = \frac{\Gamma(\alpha_i + \beta_j)}{\Gamma(\alpha_i)\Gamma(\beta_j)} p_{ij}^{\alpha_i - 1}(1 - p_{ij})^{\beta_j - 1} \tag{14}$$

where $\Gamma(x) = (x-1)!$ is the Gamma function. In other words, p_{ij}, modeling the probability of v_i selling products to v_j, obeys the Beta distribution with the parameter (α_i, β_j).

Figure 9 shows the probability density function of p_{ij} for different combinations of (α_i, β_j) when $\alpha_i > 1$,$\beta_j > 1$. We can see that for large α_i and small β_j, p_{ij} is more probable to be close to 1, indicating a high probability of cheating behavior from v_i to v_j. On the other hand, if α_i is small and β_j is large, p_{ij} tends to be small, leading to a low probability of cheating from v_i to v_j. It is intuitively reasonable since cheating is more likely to happen in pairs if one has high probability of being a cheating seller and the other has high probability to be a cheating buyer.

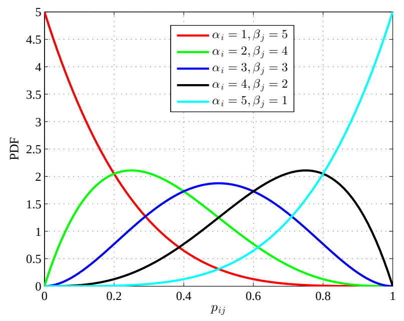


Figure 9: The probability density function of p_{ij} for different combinations of $(\alpha_i$,$\beta_j)$

To fit the graph data, we would like to choose the proper latent variables (i.e. α and β) for each node so that it is the most probable that w_{ij} can be generated by $\boldsymbol{\theta}$. In other words, $p(w_{ij}|\boldsymbol{\theta})$ in Equation (13) can be defined as follows,

$$p(w_{ij}|\boldsymbol{\theta}) = \mathtt{Beta}(w_{ij}; \alpha_i, \beta_j) = \frac{\Gamma(\alpha_i + \beta_j)}{\Gamma(\alpha_i)\Gamma(\beta_j)} w_{ij}^{\alpha_i - 1}(1 - w_{ij})^{\beta_j - 1} \tag{15}$$

The design of $\mathtt{C}(\boldsymbol{\theta})$. To avoid the over fitting, we constrain the update of the model parameters from the original ones. Formally, let α' and β' denote the initial values for all the nodes before training. This term is thus designed as

$$\mathtt{C}(\boldsymbol{\theta}) = \sum_{v_i \in V} D_{KL}(\alpha_i, \beta_i; \alpha', \beta') \tag{16}$$

It actually limits the KL Divergence between the two Beta functions of final parameter values and the initial ones. It controls the modification on the parameters cannot go far.

We can further compute $D_{KL}(\alpha_i, \beta_i; \alpha', \beta')$ as follows,

$$\begin{aligned} & D_{KL}(\alpha_i, \beta_i; \alpha', \beta') \\ &= \int_0^1 f(x; \alpha', \beta') \cdot \ln \frac{f(x; \alpha', \beta')}{f(x; \alpha_i, \beta_i)} dx \\ &= \int_0^1 f(x; \alpha', \beta') \ln f(x; \alpha', \beta') dx - \int_0^1 f(x; \alpha', \beta') \ln f(x; \alpha_i, \beta_i) dx \\ &= \ln \frac{\mathrm{B}(\alpha_i, \beta_i)}{\mathrm{B}(\alpha', \beta')} + (\alpha' - \alpha_i)\Psi(\alpha') + (\beta' - \beta_i)\Psi(\beta') \\ &\quad + (\alpha_i - \alpha' + \beta_i - \beta')\Psi(\alpha' + \beta') \end{aligned} \tag{17}$$

where $\Psi(x)$ is the digamma function, $\mathrm{B}(\cdot)$ is the beta function and $f(x; \alpha, \beta)$ is the probability density function of $\mathrm{Beta}(\alpha, \beta)$.

Model learning. To learn the parameters to minimize the objective function in Equation (13) we use the gradient descent method, shown in Algorithm 3. The general idea is to first assign the initial values to the parameters and compute the partial derivatives for them. Then, at each iteration the parameters are updated based on the partial derivatives until convergence. After some empirical studies we find that the change of the initial values α' and β' do not change the final ranking order of partners. Thus, we set $\alpha' = 10, \beta' = 10$ in this study.

The partial derivatives is shown as follows,

$$\frac{\partial \mathcal{L}}{\partial \alpha_i} = -\sum_{(v_i,v_j)\in E} \frac{\partial \log p(w_{ij}|\boldsymbol{\theta})}{\partial \alpha_i} + \lambda \cdot \frac{\partial \mathtt{C}}{\partial \alpha_i} \tag{18}$$

$$\frac{\partial \mathcal{L}}{\partial \beta_j} = -\sum_{(v_i,v_j)\in E} \frac{\partial \log p(w_{ij}|\boldsymbol{\theta})}{\partial \beta_j} + \lambda \cdot \frac{\partial \mathtt{C}}{\partial \beta_j} \tag{19}$$

where

$$\frac{\partial \log p(w_{ij}|\boldsymbol{\theta})}{\partial \alpha_i} = \log w_{ij} - \Psi(\alpha_i) + \Psi(\alpha_i + \beta_j)$$

$$\frac{\partial \log p(w_{ij}|\boldsymbol{\theta})}{\partial \beta_j} = \log(1 - w_{ij}) - \Psi(\beta_j) + \Psi(\alpha_i + \beta_j)$$

$$\frac{\partial \mathtt{C}}{\partial \alpha_i} = \Psi(\alpha_i) - \Psi(\alpha') - \Psi(\alpha_i + \beta_i) + \Psi(\alpha' + \beta')$$

$$\frac{\partial \mathtt{C}}{\partial \beta_j} = \Psi(\beta_j) - \Psi(\beta') - \Psi(\alpha_j + \beta_j) + \Psi(\alpha' + \beta')$$

Algorithm 3 Parameter learning of the probabilistic model

```
Input: a directed graph $G = (V, E, w)$, the parameter $\lambda$
 1: initialize $\alpha_k := \alpha', \beta_k = \beta'$ for all nodes $v_k \in V$
 2: repeat
 3:     for all $v_k \in V$ do
 4:         initialize $\nabla\alpha_k = \lambda \cdot \frac{\partial \mathtt{C}}{\partial \alpha_k}, \nabla\beta_k = \lambda \cdot \frac{\partial \mathtt{C}}{\partial \beta_k}$
 5:     end for
 6:     for all $(v_i, v_j) \in E$ do
 7:         $\nabla\alpha_i := \nabla\alpha_i - (\log w_{ij} - \Psi(\alpha_i) + \Psi(\alpha_i + \beta_j))$
 8:         $\nabla\beta_j := \nabla\beta_j - (\log(1 - w_{ij}) - \Psi(\beta_j) + \Psi(\alpha_i + \beta_j))$
 9:     end for
10:     $\nabla\hat{\alpha} = \max_i(|\nabla\alpha_i|), \nabla\hat{\beta} = \max_j(|\nabla\beta_j|)$
11:     for all $v_k \in V$ do
12:         $\alpha_k := \alpha_k - \frac{\nabla\alpha_k}{\nabla\hat{\alpha}}$
13:         $\beta_k := \beta_k - \frac{\nabla\beta_k}{\nabla\hat{\beta}}$
14:     end for
15: until convergence
```

5.3 Ranking based on the Probabilistic Model

After we learn the latent variables of all the nodes (i.e. α_i and β_i for $i = 1, \cdots, n$), for any edge $(v_i, v_j) \in E$ we can compute the expected value of p_{ij} as follows,

$$\pi_{ij} = \frac{\alpha_i}{\alpha_i + \beta_j} \tag{20}$$

Here, π_{ij} is the expected value of the Beta distribution with the parameters of (α_i, β_j).

Ranking on the bipartite with the probabilistic model. Then, after we perform the probabilistic model on the bipartite the new ranking score can be defined as

$$score(i) = \begin{cases} \pi_{iB} - \pi_{*i}, & \text{if } v_i \in A \\ \pi_{Ai} - \pi_{i*}, & \text{if } v_i \in B \end{cases} \tag{21}$$

where $\pi_{iB} = \sum_{(v_i,v_j)\in E \wedge v_j \in B} \pi_{ij}$, $\pi_{*i} = \sum_{(v_j,v_i)\in E} \pi_{ji}$, and π_{Ai}, π_{i*} are defined similarly.

This ranking score is obtained by replacing w_{ij} in Equation (8) with π_{ij}. We call this ranking method *Cut_ProbRank* since both the graph cut method and the probabilistic model are used. The empirical experiments will show that the new ranking method with the support of the probabilistic model significantly outperform the original method based on the edge weights.

6. EXPERIMENTAL EVALUATION

In this section, we detail the experimental results on a real-world data set from a world-wide IT company.

6.1 Data Set and Evaluation Metrics

The data set for evaluation is from the market channel of a world-wide IT company. We have two kinds of partners, namely *gold membership partners* and *silver membership partners*. We have totally 104 gold membership partners and 424 silver membership partners. For each partner the sequence of its monthly purchase volume is provided. The time interval of the sequence is from January 2009 to December 2012, a total of 48 months. The following experiments are conducted on the following two groups of partners, namely gold membership partners (Gold for short), and all the partners (All for short).

Moreover, we have a blacklist of real cheating partners as the ground truth data. There are 17 (85) true cheating partners out of the gold (silver) membership partners[3]. Thus, we aim to rank the real cheating partners as high as possible by the proposed method. Thus, the proposed model will provide some guidance for the audit work and greatly reduce the manual efforts for the judicial examination of the business and financial records.

With the blacklist of M true cheating partners, we develop the following metrics to measure how high these real cheating partners are ranked by the proposed method. For a number k, we can count the number of hits on the real cheating partners in the top-k ranking list. Then, we define *precision*, *recall*, and $F1$ as follows,

$$\begin{aligned} precision@k &= \frac{\text{Number of hits in the top-}k\text{ list}}{k} \\ recall@k &= \frac{\text{Number of hits in the top-}k\text{ list}}{M} \\ F1@k &= 2 \cdot \frac{precision@k \times recall@k}{precision@k + recall@k} \end{aligned} \tag{22}$$

Clearly, the above metrics all depend on k. Thus, we can plot the curves as shown in Figure 10, where the X-axis represents k and the Y-axis shows the metric values. Then, for each of the three curves we can calculate the Area Under Curve (AUC) metric. These three values are denoted as AUC_p, AUC_r, AUC_F1 for the curves of precision, recall and F1 respectively. Note that the higher values of these three metrics indicate better performances of a ranking method.

6.2 Experiment Summary and Methods

Here, we summarize the experimental results we plan to empirically show and discuss the compared methods. Specifically, we will show the following results.

[3]Note this blacklist does not distinguish true cheating sellers from true cheating buyers.

Table 2: Summary of the ranking methods for comparison

Name	Description
PearsonRank	the undirected graph used
Noncut_Rank	the directed graph used, without the graph cut method, without the probabilistic model, Equation (23)
Noncut_ProbRank	the directed graph used, without the graph cut method, with the probabilistic model, Equation (24)
Cut_Rank	the directed graph used, with the graph cut method, without the probabilistic model, Equation (12)
Cut_ProbRank	the directed graph used, with the graph cut method, with the probabilistic model, Equation (21)

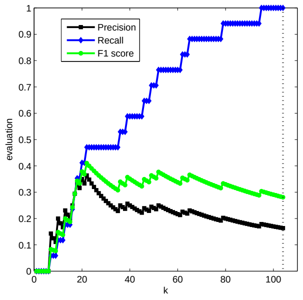


Figure 10: The example curves of precision, recall and F1 score

1) Asymmetric correlation measure. First, we will show the proposed asymmetric correlation measure is effective in this ranking task. In other words we will show that the ranking methods based on the asymmetric correlation measure are significantly better than the ones based on the symmetric correlation measure.

For this comparison we propose a ranking method based on the symmetric Pearson correlation. The computing process is as follows: 1) for each pair of partners we compute the symmetric Pearson correlation between the corresponding two sequences of sales volumes; 2) generate an undirected graph of partner correlation, where each node represents individual partner and the edge weight is set to the negative value of the Pearson correlation value between the linked two nodes. A user-specified parameter $0 < \eta < 1$ can also used here to remove the edges with the weights smaller than η; 3) for each node the ranking score is defined as the weight sum of its linked edges. We call this method *PearsonRank*.

2) Partitioning of partners for edge removing. Second, we will show that the proposed method of Max-DifCut helps to improve the ranking performance. In other words, we will empirically show that ranking on the resultant bipartite from Max-DifCut is significantly better than ranking on the original PCG graph.

For this comparison, we propose the following ranking score for any node v_i on the original directed PCG graph,

$$score(i) = |w_{i*} - w_{*i}| \qquad (23)$$

It is actually the difference of the weight sum of its out-edges and in-edges. We call this ranking method *Noncut_Rank* since there is no the graph cut method used in advance.

3) Probabilistic model to generate the graph. Third, we will show that the ranking methods based on the learned parameters of the proposed probabilistic model are significantly better than the ones based on the edge weights. We show this on both the generated bipartite and the original PCG graph.

•On the generated bipartite we actually have two ranking methods, namely *Cut_Rank* (detailed in Section 5.1) and *Cut_ProbRank* (detailed in Section 5.3). We will show that *Cut_ProbRank* is significantly better than *Cut_Rank* in terms of ranking performance.

•On the original PCG graph we already have the method of *Noncut_Rank* (in Equation (23)) based on the edge weights. Similarly, based on the probabilistic model we propose the following ranking score for any node v_i on the original directed PCG graph,

$$score(i) = |\pi_{i*} - \pi_{*i}|, \qquad (24)$$

where $\pi_{i*} = \sum_{(v_i,v_j)\in E} \pi_{ij}$, and $\pi_{*i} = \sum_{(v_j,v_i)\in E} \pi_{ji}$. This ranking score is obtained by replacing w_{ij} in Equation (23) with π_{ij}. We call this ranking method *Noncut_ProbRank* since the probabilistic model is used here, however, the graph cut method is not used. We will show that *Noncut_ProbRank* is significantly better than *Noncut_Rank* in terms of ranking performance.

For easy-reading we summarize all the ranking methods for comparison in Table 2.

4) Different Parameter Settings. Finally, we will also show that the above experimental results are consistent with different parameter settings. In our methods we have the following three parameters, i.e. the dynamic warping scope s, the threshold η (for removing those edges whose correlations are not negative enough), and the parameter λ used in the probabilistic model. The ranges for these parameters are discussed as follows.

•DTW Scope s. Through market investigation, a partner can only store the products at most for 3 months. Thus, we select s from $\{1, 2, 3\}$.

•Threshold η. we set up a threshold η to remove those edges whose correlations are not negative enough. For a big η, the resultant graph will contain the isolated nodes which do not have any in-edges and out-edges, and without any linked edges the ranking function on these isolated nodes becomes useless. Thus, we need the range of η so that the generated graph is a connected one. Table 3 shows this range of η for different settings of s on the two data sets. In this study, we select the η value by the step of 0.1.

•For probability ranking methods (i.e. *Noncut_ProbRank* and *Cut_ProbRank*), we set λ to 0.1 by default. We will also empirically show that λ affects the ranking performance of the probabilistic model.

Table 3: The parameter settings

Dataset	s	η
Gold	1	{0,0.1,0.2,0.3}
	2	{0,0.1,0.2,0.3,0.4}
	3	{0,0.1,0.2,0.3,0.4,0.5}
All	1	{0,0.1,0.2,0.3}
	2	{0,0.1,0.2,0.3,0.4}
	3	{0,0.1,0.2,0.3,0.4,0.5}

6.3 Experimental Results

Table 4: The best performance of each method

Dataset	Method	AUC		
		AUC_p	*AUC_r*	*AUC_F1*
Gold	*PearsonRank*	20.58	56.41	26.07
	Noncut_Rank	20.75	59.62	28.11
	Noncut_ProbRank	24.39	67.29	31.82
	Cut_Rank	27.40	70.91	35.22
	Cut_ProbRank	**29.11**	**71.79**	**36.56**
All	*PearsonRank*	101.00	308.36	139.70
	Noncut_Rank	82.00	259.64	111.64
	Noncut_ProbRank	106.09	295.71	135.36
	Cut_Rank	120.68	345.45	162.27
	Cut_ProbRank	**143.79**	**373.55**	**182.59**

Table 5: The average performance of each method

Dataset	Method	AUC		
		AUC_p	*AUC_r*	*AUC_F1*
Gold	*PearsonRank*	19.80	56.22	25.82
	Noncut_Rank	15.01	50.05	21.68
	Noncut_ProbRank	17.62	54.58	24.36
	Cut_Rank	23.94	64.62	31.35
	Cut_ProbRank	**24.77**	**66.31**	**32.27**
All	*PearsonRank*	99.48	303.86	137.50
	Noncut_Rank	73.47	246.79	104.26
	Noncut_ProbRank	79.27	255.49	109.78
	Cut_Rank	110.17	372.87	150.48
	Cut_ProbRank	**120.87**	**339.37**	**160.05**

Table 6: The t-test on each pair of methods

Dataset	Pairs of Methods	p-value
Gold	(*Noncut_Rank*,*Noncut_ProbRank*)	4.67×10^{-4}
	(*Cut_Rank*,*Cut_ProbRank*)	1.47×10^{-2}
	(*Noncut_Rank*,*Cut_Rank*)	1.57×10^{-7}
	(*Noncut_ProbRank*,*Cut_ProbRank*)	1.90×10^{-7}
All	(*Noncut_Rank*,*Noncut_ProbRank*)	3.33×10^{-2}
	(*Cut_Rank*,*Cut_ProbRank*)	1.52×10^{-4}
	(*Noncut_Rank*,*Cut_Rank*)	3.43×10^{-11}
	(*Noncut_ProbRank*,*Cut_ProbRank*)	2.46×10^{-9}

Recall that the ranges of s and η for the two data sets are shown in Table 3. On each setting of s and η we can compute the values of AUC_p, AUC_r, and AUC_F1 for each method. Then, for each method the best value and the average value over all the parameter settings are shown in Tables 4 and 5. For each pair of the ranking methods on the directed graph we also perform the t-test to check whether the performance difference of the corresponding two methods is statistically significant. The p-values for each pair of methods are shown in Table 6. With these three tables we have the following findings:

• The ranking of the four methods on the directed graph is *Cut_ProbRank* > *Cut_Rank* > *Noncut_ProbRank* > *Noncut_Rank* in terms of the best performance and the average performance. Note that as shown in Table 6 this performance ranking is statistically significant when the p-value threshold is set to 0.05. Also, this ranking holds on the two data sets of Gold and All. As a whole, *Cut_ProbRank* with the graph cut method and the probabilistic model outputs the best performance.

• In terms of both the best performance and the average performance the methods of *Cut_ProbRank* and *Cut_Rank* are better than *PearsonRank* on both the two data sets of Gold and All[4]. It indicates that the asymmetric correlation measure helps to improve the ranking performance.

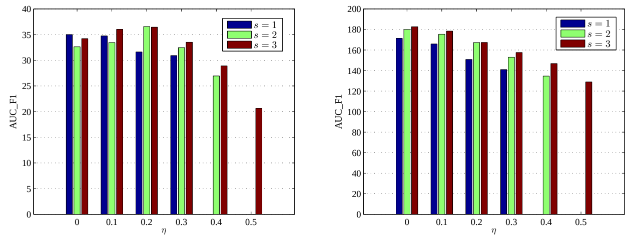
• We can see that: the ranking on the resultant bipartite is always better than the one on the original PCG graph (*Cut_ProbRank* > *Noncut_ProbRank* and *Cut_Rank* > *Noncut_Rank*). It indicates that the proposed method of Max-DifCut removes some noisy edges on the original PCG graph, and then helps to improve the ranking performance.

• Additionally, we can also see that the ranking based on the probabilistic model is always better than the one based on the edge weights (*Cut_ProbRank* > *Cut_Rank* and *Noncut_ProbRank* > *Noncut_Rank*). It indicates that the proposed probabilistic model helps to further improve the ranking performance. The new ranking method is based on the values of π_{ij} on any edge from v_i to v_j. The value of π_{ij} is actually the adjustment of the corresponding edge weight w_{ij}. These empirical experiments show this adjustment significantly improves the ranking performance.

• For each method with the parameters for the best performance we also plot its curves of precision, recall, and F1 along the the increase of k. These curves are shown in Figure 11. Check the Figures 11a, 11b, 11c for the data set All (the figures for the data set Gold show the similar results). We can clearly see that the curves for the method *Cut_ProbRank* are always at the topmost. As shown in Figure 11a, the precision of *Cut_ProbRank* at $k = 30$ is around 60%. It means that more than half of the partners in its top-30 ranking list are true cheating partners. This is really a great achievement considering that our method is totally unsupervised. For the other methods the precision values are all below 30%.

Moreover, we check how the parameters of s and η affect the ranking performance of *Cut_ProbRank* method. As shown in Figure 12, the increase of s brings better ranking performance. Since bigger s allows larger time delay when computing the DPC value, it indicates that cheating behaviors may happen with three-month time delay. Also, Figure 12 shows that the smaller η generates better ranking performance. This may indicate that the edges with the small weight values are still important for this ranking task.

[4]Since the method of *PearsonRank* has only one parameter η we do not perform the t-test between this method and the other ones.



(a) on the data set of Gold **(b) on the data set of All**

Figure 12: Evaluation on the impact of s and η for *Cut_ProbRank*

Finally, Figure 13 shows how parameter λ affects the ranking performance of *Cut_ProbRank* and *Noncut_ProbRank*, the two probabilistic methods. For *Cut_ProbRank*, the increase of λ slightly decreases its ranking performance. We explain this observation as follows. *Cut_ProbRank* performs the probabilistic model on the bipartite after removing the noisy edges by the graph cut method. Since the edges remaining in the bipartite are quite clean, completely fitting the data with $\lambda = 0$ outputs the best performance. On the contrary, *Noncut_ProbRank* performs to fit all the edges in the original PCG graph without removing the noisy edges. At this time, the increase of λ helps to constrain the change of the model parameters, and thus very slightly improve the ranking performance.

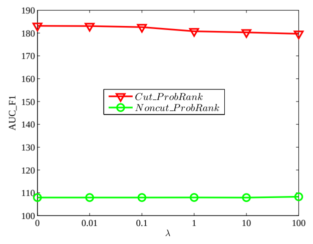
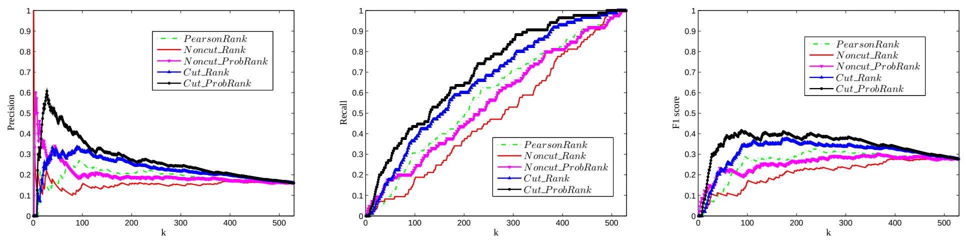


Figure 13: Evaluation on the impact of λ for *Cut_ProbRank* and *Noncut_ProbRank* when $s = 3$ and $\eta = 0$

7. RELATED WORK

In this section, we review the related work to this study. First, we list some previous works on applying Pearson correlation and dynamic time warping to real world applications. Then, we discuss the previous works on the graph cut methods.

Pearson correlation. Pearson correlation [12] is a well-known measure of linear correlation of two dependent variables. This method was first developed by Karl Pearson and is widely used to measure the similarity between pair-wise time series. Liao [21] used this measure for clustering time series. Papadimitriou et al. [17] proposed a local correlation measure to track the correlation among time-evolving time series. Xiong et al. [22] explored the upper bound of Pearson correlation to identify strongly correlated pair-wise time sequences. Kawale et al. [7] used the correlation measure to discover dipoles, which represent long distance connections between the pressure anomalies of two distant regions that are negatively correlated with each other. Their contribution is to propose the method to evaluate the significance of the correlations. We also tried this significance measure in our application. However, the experiments show that removing the correlations with small significance values does not clearly improve the ranking performance for our task. Note that all the correlation measures used in these studies are symmetric.



(a) Precision on All **(b) Recall on All** **(c) F1 on All**

Figure 11: The evaluation curves of the best performance of each method

Dynamic time warping. Dynamic time warping is widely used in various fields such as bioinfomatics, chemical engineering and robotics etc. [1, 11, 15, 19]. Keogh et al. [8] applied DTW technique for exactly indexing time series from large database by proposing a lower bound of DTW distance. In [9] and [10], DTW is modified to approximate high level abstraction of data and track the local accelerations and decelerations in the time axis.

By introducing DTW technique with the forward warping direction to Pearson correlation, we develop the measure of Directed Pearson Correlation. The forward warping direction is motivated by the fact that the cheating deals between cheating sellers and cheating buyers cannot be completed at the same month of the falsified deals, but with a time delay up to 3 months. This directed measure helps to differentiate cheating sellers and cheating buyers, and improve the ranking performance as shown by the empirical experiments.

Graph cut methods. In this study, motivated by the application background of cheating detecting we propose a new graph cut problem, called Max-DifCut, with the objective function of $3w_{AB} - w_{BA}$ (detailed in Equation (10)). Most of the studies in this area focus on the Max-DiCut problem with the objective function of w_{AB} and propose the greedy and SDP solutions [3, 4, 5, 14]. We also tried the SDP solution to the new problem. The SDP solution achieves the bipartite with the bigger value of the objective function. However, the ranking on the bipartite from the SDP solution does not improve clearly. Thus, in this paper we only discuss the greedy solution to this new problem.

8. CONCLUSION AND FUTURE WORK

In this paper we address the problem of detecting cheating in the distribution channels. Our solution consists of three modules: 1) use the developed directed Pearson correlation measure to build the directed Partner Correlation Graph; 2) Partitioning the graph by the Max-DifCut task; 3) Ranking based on the probabilistic model, which fits the bipartite graph (generated by the second step). The experiments show that the method *Cut_ProbRank* with all these modules achieves the best performance with the statistical significance in ranking the cheating partners.

The current work only considers the increase or decrease tendency in computing the correlation of two sales-volume sequences. In the future we will consider the quantities each partner buys from the manufacturer to further improve the performance. Also, we will further explore the situation that the character of a partner can change with time. At last, the proposed pairwise correlation measure may not work well when the groups of partners collude in the distribution channel. We will try to combine the Granger Causality [13] with current DPC to address this problem.

9. ACKNOWLEDGEMENTS

This work was supported by the National Natural Science Foundation of China (No. 61175052, 61203297, 61035003, 61473274), National High-tech R&D Program of China (863 Program) (No. 2014AA015105, 2013AA01A606, 2012AA011003) and Chongqing Science Foundation (No. CSTC 2010BB2214).

10. REFERENCES

[1] John Aach and George M Church. Aligning gene expression time series with time warping algorithms. *Bioinformatics*, 17(6):495–508, 2001.

[2] Michael R Garey and David S Johnson. Computer and intractability: A guide to the np-completeness. *WH Freeman and Company*, 1979.

[3] Michel X Goemans and David P Williamson. 0.879-approximation algorithms for max cut and max 2sat. In *STOC*, 1994.

[4] Michel X Goemans and David P Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM*, 42(6):1115–1145, 1995.

[5] Eran Halperin and Uri Zwick. Combinatorial approximation algorithms for the maximum directed cut problem. In *SODA*, 2001.

[6] Kenneth Hardy and Allan Magrath. Dealing with cheating in distribution. *European Journal of Marketing*, 23(2):123–129, 1989.

[7] Jaya Kawale, Snigdhansu Chatterjee, Dominick Ormsby, Karsten Steinhaeuser, Stefan Liess, and Vipin Kumar. Testing the significance of spatio-temporal teleconnection patterns. In *KDD*, 2012.

[8] Eamonn Keogh and Chotirat Ann Ratanamahatana. Exact indexing of dynamic time warping. *Knowledge and information systems*, 7(3):358–386, 2005.

[9] Eamonn J Keogh and Michael J Pazzani. Scaling up dynamic time warping for data mining applications. In *KDD*, 2000.

[10] Eamonn J Keogh and Michael J Pazzani. Derivative dynamic time warping. In *SDM*, 2001.

[11] Zsolt Miklos Kovacs Vajna. A fingerprint verification system based on triangular matching and dynamic time warping. *TPAMI*, 22(11):1266–1276, 2000.

[12] Joseph Lee Rodgers and W Alan Nicewander. Thirteen ways to look at the correlation coefficient. *The American Statistician*, 42(1):59–66, 1988.

[13] Aurélie C Lozano, Naoki Abe, Yan Liu, and Saharon Rosset. Grouped graphical granger modeling for gene expression regulatory networks discovery. In *KDD*, 2009.

[14] Shiro Matuura and Tomomi Matsui. 0.863-approximation algorithm for max dicut. In *APPROX*, 2001.

[15] Mario E Munich and Pietro Perona. Continuous dynamic time warping for translation-invariant curve alignment with applications to signature verification. In *ICCV*, 1999.

[16] Das Narayandas and V Kasturi Rangan. Building and sustaining buyer-seller relationships in mature industrial markets. *Journal of Marketing*, 68(3):63–77, 2004.

[17] Spiros Papadimitriou, Jimeng Sun, and Philip S Yu. Local correlation tracking in time series. In *ICDM*, 2006.

[18] Hiroaki Sakoe and Seibi Chiba. Dynamic programming algorithm optimization for spoken word recognition. *Acoustics, Speech and Signal Processing*, 26(1):43–49, 1978.

[19] M Schmill, Tim Oates, and P Cohen. Learned models for continuous planning. In *Proceedings of Uncertainty*, 1999.

[20] TK Vintsyuk. Speech discrimination by dynamic programming. *Cybernetics and Systems Analysis*, 4(1):52–57, 1968.

[21] T Warren Liao. Clustering of time series data:a survey. *Pattern Recognition*, 38(11):1857–1874, 2005.

[22] Hui Xiong, Shashi Shekhar, Pang-Ning Tan, and Vipin Kumar. Exploiting a support-based upper bound of pearson's correlation coefficient for efficiently identifying strongly correlated pairs. In *KDD*, 2004.