Exploring Correlation Network for Cheating Detection

PING LUO, Key Laboratory of Intelligent Information Processing, Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China; University of Chinese Academy of Sciences, Beijing 100049, China
KAI SHU, Department of Computer Science and Engineering, Arizona State University, Tempe, AZ, USA
JUNJIE WU, School of Economics and Management, Beihang University, Beijing, China; Beijing Advanced Innovation Center for Big Data and Brain Computing, Beihang University, Beijing, China
LI WAN, Department of Computer Science and Technology, Chongqing University, Chongqing, China
YONG TAN, Department of Information Systems and Operations Management, University of Washington, Seattle, WA, USA

The correlation network, typically formed by computing pairwise correlations between variables, has recently become a competitive paradigm to discover insights in various application domains, such as climate prediction, financial marketing, and bioinformatics. In this study, we adopt this paradigm to detect cheating behavior hidden in business distribution channels, where falsified big deals are often made by collusive partners to obtain lower product prices—a behavior deemed to be extremely harmful to the sale ecosystem. To this end, we assume that abnormal deals are likely to occur between two partners if their purchase-volume sequences have a strong negative correlation. This seemingly intuitive rule, however, imposes several research challenges. First, existing correlation measures are usually symmetric and thus cannot distinguish the different roles of partners in cheating. Second, the tick-to-tick correspondence between two sequences might be violated due to the possible delay of purchase behavior, which should also be captured by correlation measures. Finally, the fact that any pair of sequences could be correlated may result in a number of false-positive cheating pairs, which need to be corrected in a systematic manner. To address these issues, we propose a correlation network analysis framework for cheating detection. In the framework, we adopt an asymmetric correlation measure to distinguish the two roles, namely, *cheating seller* and *cheating buyer*, in a cheating alliance. *Dynamic Time Warping* is employed to address the time offset between two sequences in computing

Dr. Ping Luo was partially supported by the National Key Research and Development Program of China under Grant 2017YFB1002104, the National Natural Science Foundation of China (NSFC) under Grant U1811461, and the Innovation Program of Institute of Computing Technology, CAS. Dr. Junjie Wu was partially supported by the National Key R&D Program of China under Grant 2019YFB2101804, and by NSFC under Grants 71725002, 71531001, U1636210, 71471009, and 71490723. Dr. Yong Tan was supported in part by NSFC under Grant 71729001.
Authors' addresses: P. Luo, Key Laboratory of Intelligent Information Processing, Institute of Computing Technology, Chinese Academy of Sciences, No. 6 Kexueyuan South Road Zhongguancun, Haidian District, Beijing 100190, China and University of Chinese Academy of Sciences, Beijing 100049, China; email: luop@ict.ac.cn; K. Shu, Department of Computer Science and Engineering, Arizona State University, 561BB Brickyard Suit, 699 S Mill Ave, Tempe, AZ 85281, USA; email: kai.shu@asu.edu; J. Wu (corresponding author), School of Economics and Management, Beihang University, No. 37 Xue Yuan Road, Haidian District, Beijing 100191, China and Beijing Advanced Innovation Center for Big Data and Brain Computing, Beihang University, No. 37 Xue Yuan Road, Haidian District, Beijing 100191, China; email: wujj@buaa.edu.cn; L. Wan, Department of Computer Science and Technology, Chongqing University, No. 174 ShaZheng Road, Shapingba District, Chongqing 400034, China; email: wanli@cqu.edu.cn; Y. Tan, Department of Information Systems and Operations Management, University of Washington, 1410 NE Campus Parkway, Seattle, Washington 98195, USA; email: ytan@uw.edu.

2157-6904/2020/01-ART12 $15.00
https://doi.org/10.1145/3364221

the correlation. We further propose two *graph-cut* methods to convert the correlation network into a bipartite graph to rank cheating partners, which simultaneously helps to remove false-positive correlation pairs. Based on a 4-year real-world channel dataset from a worldwide IT company, we demonstrate the effectiveness of the proposed method in comparison to competitive baseline methods.

CCS Concepts: • **Applied computing** → **Business intelligence**; • **Information systems** → **Data mining**;

Additional Key Words and Phrases: Correlation network analysis, cheating detection, distribution channel, time series, graph cut

ACM Reference format:
Ping Luo, Kai Shu, Junjie Wu, Li Wan, and Yong Tan. 2020. Exploring Correlation Network for Cheating Detection. *ACM Trans. Intell. Syst. Technol.* 11, 1, Article 12 (January 2020), 23 pages.
https://doi.org/10.1145/3364221

1 INTRODUCTION

Philosophically, all things on Earth are correlated with each other in a systematic way. This idea corresponds with the method paradigm *correlation network building and analysis*, as shown in Figure 1, which has recently gained increasing research interest in various application domains [Berezin et al. 2012; Kenett et al. 2010; Ludescher et al. 2014; Tumminello et al. 2010]. In this task, we are given a set of items $V = \{v_i\}_{i=1}^n$ with their time series $\{\vec{x}_i\}_{i=1}^n$, where $\vec{x}_i$ is the sequence attached to item v_i. Then, the correlation network can be represented as a weighted graph over V. For each pair of items $v_i, v_j \in V$, a weight w_{ij} can be calculated by measuring the correlation between the two corresponding time series $\vec{x}_i$ and $\vec{x}_j$. The real world contains many examples of correlation networks. For example, in climate science research, a node often refers to a specific location, and its time series refers to the surface air temperatures over time at that location. In finance engineering research, a node usually refers to a specific stock, and its time series refers to the corresponding price curve over time.

After building a correlation network, analysis of the network can be conducted to solve real-life problems rising in different domains, such as financial marketing [Bonanno et al. 2004; Kenett et al. 2010; Mantegna 1999; Tumminello et al. 2010], climate science [Berezin et al. 2012; Donges et al. 2009; Ludescher et al. 2014], and bioinformatics [Bar-Joseph 2004; Steuer et al. 2003]. Correlation-based networks were first introduced in Mantegna [1999] for financial market studies, which found that the topological arrangement of the U.S. stock correlation network could reflect the structure of industry sectors. Bonanno et al. [2004] showed that the correlation graph shifted from a structured and clustered graph to a simple starlike graph with a decrease of the time interval between any two successive entries in the time series. Tumminello et al. [2010] discussed methods for quantitatively exploring the correlations among equities and building correlation graphs. Kenett et al. [2010] studied the dynamics of stock market correlations by visualizing the correlation graph in three-dimensional (3D) space over time. Some recent studies applied correlation networks in climate science. For instance, Donges et al. [2009] identified the *El Niño basin*, an area in the Pacific Ocean where surface temperatures are highly correlated with the temperatures at many other locations. They also found that the coming of the El Niño climate actually disrupts correlations between temperatures at different locations worldwide. Furthermore, Ludescher et al. [2014] leveraged this climate network for the prediction of El Niño arrival more than 6 months ahead of time.

Motivated by the above successful applications, in this article, we study how to adapt the correlation network analysis paradigm to detect cheating behavior in business distribution channels. In what follows, we briefly describe the background knowledge of the distribution channel and the typical cheating behaviors.

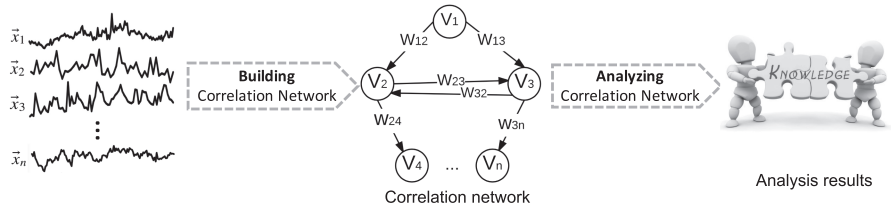
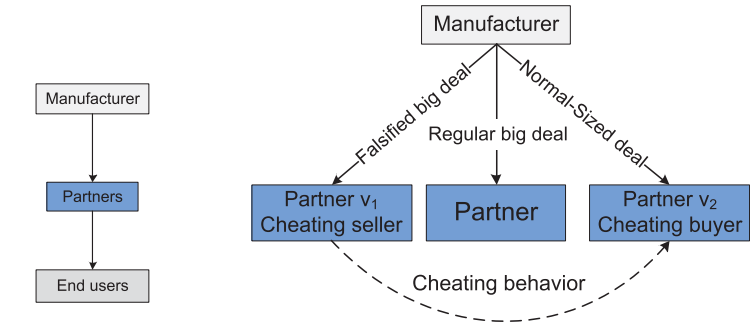


Fig. 1. Method paradigm of correlation network building and analysis.



(a) Distribution Channel (b) Cheating Behavior

Fig. 2. Cheating behavior in the distribution channel.

1.1 Cheating Behavior in Distribution Channel

As shown in Figure 2(a), a distribution channel is a system of *partners* to move products from the *manufacturer* to *end users*. This indirect channel is extremely helpful to improving product revenue and market competitiveness when the number of end users is too large to be covered by the manufacturer directly. Due to its importance, manufacturers need to develop effective channel policies, especially for price management, to maintain a smoothly operating distribution channel.

To spur the sales enthusiasm of channel partners, it is common for a manufacturer to adjust product prices according to the sales volume of a deal. If a partner plans to buy a large volume of products due to sufficient demand, then she can apply for a low price from the manufacturer. On the contrary, for a normal-sized deal, the partner can only get the regular price. Hence, the price differs for different partners. This pricing policy, however, is like a double-edged sword that may also breed cheating in the distribution channel.

A typical scenario of a *cheating alliance* is shown in Figure 2(b). To earn a low price, partner v_1 orders a falsified large deal from the manufacturer and re-sells some of the product to another partner v_2 at a price lower than the regular price of a normal-sized deal. In this scenario, we call v_1 and v_2 the *cheating seller* and *cheating buyer*, respectively, who actually damage the ecosystem of the distribution channel and must be detected.

1.2 Problem Formulation

To detect cheating behavior, a manufacturing company often builds a specialized team of audit staff to monitor abnormal activity in the distribution channel. Their everyday work involves the manual examination of business records and even judicial investigations for severe cheating cases. To guide the tedious audit process and reduce manual efforts, in this study, we aim to provide an automatic method for cheating detection.

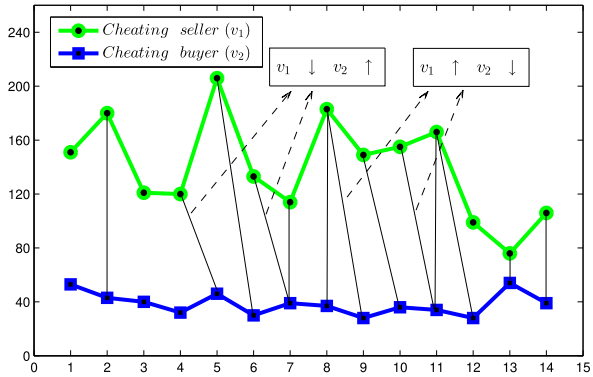


Fig. 3. Time series of the purchase quantities of two real cheating partners.

Our problem can be described as follows. We are given a set $V = \{v_1, \ldots, v_n\}$ of n partners. For each partner v_i, we also know its purchase-volume time series $\vec{x}_i$, where each entry records the volume of product v_i purchased from the manufacturer during a specific time interval (1 month in this study). With these time-series data for all the partners, we aim to rank the partners by the degree of their suspect cheating behavior, either as a seller or buyer.

Some previous works have discussed channel cheating from a qualitative perspective. Hardy and Magrath [1989] summarized the types and forms of channel cheating and then explored the structures, factors, and incentives that encourage cheating. Narayandas and Rangan [2004] 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 is the behavior of partner alliances. To the best of our knowledge, our work is among the first quantitative studies on cheating detection in distribution channels.

It is worth noting that our focus here is to identify cheating suspects rather than to obtain a guilty verdict. With the ranked list of cheating suspects, audit staff can plan their everyday work more efficiently and pay more attention to the partners ranked higher in the list. The final determination of true cheating partners is subject to further legal investigation to obtain proof.

1.3 Motivating Observations and Contributions

Our approach is based mainly on the correlation analysis of the purchase-volume time series between partners. Figure 3 illustrates the time series of the purchase volumes of two real-world cheating partners, where v_1 and v_2 are the paired cheating seller and cheating buyer, respectively. Their sales volumes exhibit interesting interactions over time:

- When v_1 applies for a falsified big deal from the manufacturer, its purchase volume increases in that month. Meanwhile, since v_2 buys some products from v_1 after the deal, its purchase volume from the manufacturer tends to decrease accordingly. For example, in Figure 3, v_1's purchase volume increases in the fifth month while v_2's purchase volume decreases in the sixth month.
- By contrast, if v_1 does not apply for big deals, then its purchase volume decreases, and the purchase volume of v_2 from the manufacturer tends to increase if with stable demand from end users. For example, v_1's purchase volume decreases in the seventh month while v_2's purchase volume increases in the seventh month.

Intuitively, this phenomenon could be treated as a clue for cheating behavior. The higher the frequency of the phenomenon, the stronger the evidence would be. Along this line, we can explore

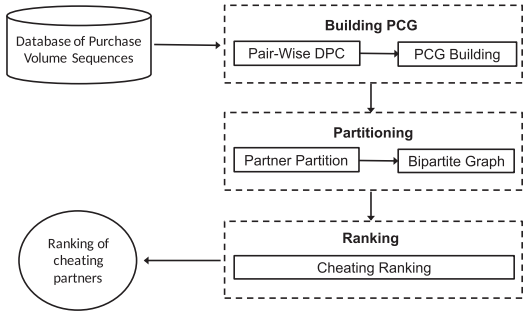


Fig. 4. Framework of cheating detection.

the correlations among the purchase-volume time series for all the partners and then adapt the paradigm of building and analyzing correlation networks for cheating detection.

This scheme, though logically clear, imposes several research challenges. First, existing correlation measures are usually symmetric and cannot distinguish the different roles of partners in cheating. Second, the implemented correlation measure should also account for the time offset of the two time series due to the possible delay of purchase behavior on one side. Finally, the fact that any pair of sequences could be correlated may result in a number of false-positive cheating pairs, which should be removed in a systematic manner.

To address these issues, we propose a correlation network analysis framework for cheating detection. In the framework, we first adopt an asymmetric correlation measure to distinguish the two roles, namely, *cheating seller* and *cheating buyer*, in the cheating alliance, which helps to build a directed network containing the correlations among partners. Dynamic Time Warping is also employed to address the time offset between two sequences when computing the correlation. By assuming a stable role for each partner over time, we propose a *graph-cut* method to convert the correlation network into a bipartite graph to obtain the final ranking of cheating partners, which could simultaneously help to remove false-positive correlation pairs. Based on a 4-year real-life channel dataset from a worldwide IT company, we empirically show the effectiveness of the proposed method compared with baseline methods. Our study indeed obtains a promising solution for real-world applications of cheating detection.

The rest of this article is organized as follows. Section 2 briefly introduces the framework of cheating detection based on correlation network analysis. In Section 3, we develop a directed correlation measure by incorporating dynamic time warping into the Pearson correlation. Section 4 details the graph-cut method for systematically removing false-positive correlation pairs. Section 5 presents the ranking method. We empirically validate the effectiveness of the proposed method in Section 6. Section 7 discusses the related work, followed by the final conclusion in Section 8.

2 FRAMEWORK FOR CHEATING DETECTION

In this section, we present the framework for cheating detection. As shown in Figure 4, the framework consists of three main steps: (1) building a partner correlation graph as the correlation network; (2) converting the graph into a bipartite graph with suspicious sellers and buyers, respectively; (3) ranking the partners by their cheating degrees. Next, we detail these three steps one by one. We describe why each step is needed and the associated challenges.

2.1 Building Partner Correlation Graph

Motivated by the observations from Figure 3, we need a method to measure the correlation between two purchase-volume sequences. If two sequences have a strong negative correlation, then it is

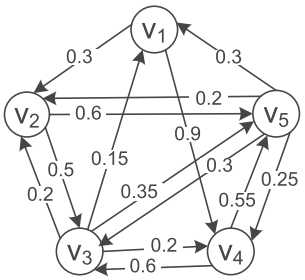


Fig. 5. An example of a partner correlation graph.

likely that there exist abnormal deals. Pearson's correlation coefficient (denoted as r) was believed to be an ideal measure for this type of task, but it actually has certain limitations in this application scenario.

First, r is a symmetric measure that cannot distinguish cheating sellers from cheating buyers. Moreover, r is usually computed with the tick-to-tick correspondence of the two sequences. However, an abnormal deal might not be completed in a single month but rather with a delay of several months after the cheating seller applies for a big deal from the manufacturer. Thus, we need to consider the time offset in the two sequences when computing the correlation.

We first leverage the *Dynamic Time Warping* (DTW) technique to address the time offset problem. DTW is a widely adopted method to find an optimal time alignment between two sequences under certain constraints. In an abnormal deal, the cheating seller can sell products to the cheating buyer only at or after the time when the cheating seller obtains a falsified big deal. Thus, the warping direction must be *from now to the future* within a time window.

Based on DTW, we then adopt an asymmetric measure called the *directed Pearson correlation* (DPC for short, denoted as r_{dpc}) to compute the correlation of two sequences $\vec{x}_1$ and $\vec{x}_2$. $r_{dpc}(\vec{x}_1, \vec{x}_2)$ 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 of how to compute r_{dpc} using DTW are presented in Section 3.

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

- $V = \{v_1, \ldots, v_n\}$ contains the n nodes, with v_i being the partner with the sequence $\vec{x}_i$;
- $E = \{(v_i, v_j) | w_{ij} > \eta, w_{ij} \in w\}$, with $\eta \in (0, 1)$ being a user-specified parameter;
- $w_{ij} = -r_{dpc}(\vec{x}_i, \vec{x}_j)$ is the weight on the edge of (v_i, v_j), $w_{ij} \in w$.

Remember that we aim to identify 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 weights exceeding η are retained in the graph. We call this graph the *partner correlation graph* (PCG). Figure 5 shows an example of a PCG with five partners when $\eta = 0.1$. We use this graph as a running example throughout this article.

2.2 Graph Partitioning for Noise Edges Removal

As shown in Figure 5, a PCG is actually a two-way directed graph, which implies that each partner can be viewed as both a cheating seller and a cheating buyer. For example, when we consider the edge from v_5 to v_1 in Figure 5, v_1 acts as a cheating buyer. Meanwhile, given the edge from v_1 to v_4, v_1 also acts as a cheating seller.

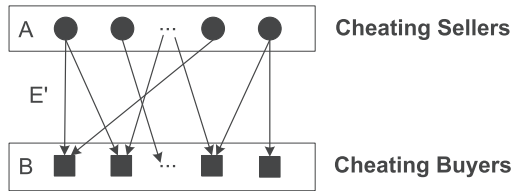


Fig. 6. A bipartite graph of candidate cheating partners.

In the real world, however, only partners with abundant capital can be cheating sellers, since a large investment is required for big deals. Additionally, these partners have no incentive to be cheating buyers. However, partners with less capital can only be cheating buyers. Therefore, the two roles are most often exclusive, that is, each partner can be either a cheating seller or a cheating buyer but not both (we also assume that the character of a partner does not change over time). Our goal is thus to differentiate suspicious cheating sellers from buyers by removing noise edges in the PCG.

The one-role assumption above means that in the PCG, the nodes of cheating sellers can have only out-edges while the nodes of cheating buyers can have only in-edges. Hence, we need a graph cutting method to change a weighted directed graph $G = (V, E, w)$ into a bipartite graph. Figure 6 gives an example of the resultant bipartite graph, where the node set A contains all the cheating sellers with only out-edges, the set B contains all the cheating buyers with only in-edges, and only the edges from A to B are retained in the graph.

The graph-cut algorithm that transforms a graph into a bipartite graph aims to maximize a specified objective function defined on G. In this study, to facilitate the subsequent ranking of cheating partners, we formulate a graph-cut problem with a new objective function. Because it is an NP-hard problem, we develop two solutions, namely, the greedy method and the semi-definite programming (SDP) method. We also discuss the performance of these two methods for ranking cheating partners. All the details are presented in Section 4.

2.3 Ranking Partners

Next, we present the measure for ranking partners. We consider the ranking measure for three types of correlation networks: (1) the undirected correlation network, where the correlation is computed via the traditional Pearson correlation coefficient; (2) the directed correlation network, where the correlation is computed via the proposed asymmetric measure; and (3) the directed bipartite graph after the graph-cut step. In particular, for the undirected correlation network, the ranking score of each node is defined as the weight sum of its linked edges; for the directed correlation network, the ranking score is the weigh sums of differences between the out-edges and in-edges; for the directed bipartite graph, we compute the ranking score of cheating sellers and cheating buyers with their weight sums of out-edges and in-edges, respectively. All these ranking measures are detailed in Section 5, and their empirical comparison is discussed in Section 6.

3 DIRECTED PEARSON CORRELATION

In this section, we propose the directed Pearson correlation for building a partner correlation graph. First, we briefly introduce the Pearson correlation and Dynamic Time Warping. We then show how to compute DPC with DTW. Let $\vec{x} = (\vec{x}(1), \vec{x}(2), \ldots, \vec{x}(m))^\top$ and $\vec{y} = (\vec{y}(1), \vec{y}(2), \ldots, \vec{y}(m))^\top$ be the two purchase volume sequences of the two partners in the past m months.

3.1 Pearson Correlation

Pearson's correlation coefficient is a widely used statistic for analyzing the strength of a relationship between two numerical variables in a form such as a time series. That is, given two sequences $\vec{x}$ and $\vec{y}$ of size m, the Pearson correlation can be computed as:

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

where $\vec{x}(t)$ ($\vec{y}(t)$) is the tth entry of $\vec{x}$ ($\vec{y}$), $\overline{x}$ ($\overline{y}$) 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 $\frac{\vec{x}(t)-\overline{x}}{\delta_{\vec{x}}}$ by $\frac{\vec{y}(t)-\overline{y}}{\delta_{\vec{y}}}$ at the same time stamp t. Moreover, the correlation is symmetric for $r(\vec{x}, \vec{y}) = r(\vec{y}, \vec{x})$ and ranges from -1 to 1, with $r = 1$ and $r = -1$ indicating perfect positive and negative linear relationships, respectively.

3.2 Dynamic Time Warping

We now briefly introduce the technique of DTW [Vintsyuk 1968], which helps to transform Pearson's correlation coefficient into an asymmetric measure. DTW is a transformation that allows sequences to be stretched along the time axis to minimize the distance between them. Given two sequences $\vec{x}$ and $\vec{y}$ of size m, a *local cost measure* $c(\vec{x}(i), \vec{y}(j))$ is defined to characterize the distance between $\vec{x}(i)$ and $\vec{y}(j)$). A *cost matrix* $C \in \mathbb{R}^{m \times m}$ can then be defined with $C(i, j) = c(\vec{x}(i), \vec{y}(j))$. Here, an alignment is a warping path $p = (p_1, \ldots, p_L)^\top$ with $p_l = (i_l, j_l) \in [1, \ldots, m] \times [1, \ldots, m]$, $l \in [1, \ldots, L]$, $L \in [m, 2m-1]$, satisfying the following three conditions:

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

The total cost of a warping path p is defined as:

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

An *optimal warping path* between $\vec{x}$ and $\vec{y}$ is a warping path p^* that has the minimal total cost:

$$c_{p^*}(\vec{x}, \vec{y}) := \min_{p} \{c_p(\vec{x}, \vec{y}) \mid p \text{ is a warping path}\}. \tag{3}$$

A dynamic programming algorithm is often employed to determine the optimal warping path. Let $d(i, j)$ denote the accumulative cost of $c_{p^*}(\vec{x}, \vec{y})$ up to the subscript pair (i, j). Then, the iterative condition holds:

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

3.3 Directed Pearson Correlation

We now extend Pearson's correlation coefficient to a novel asymmetric correlation measure by using the DTW method. To that end, we first define the following local cost measure:

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

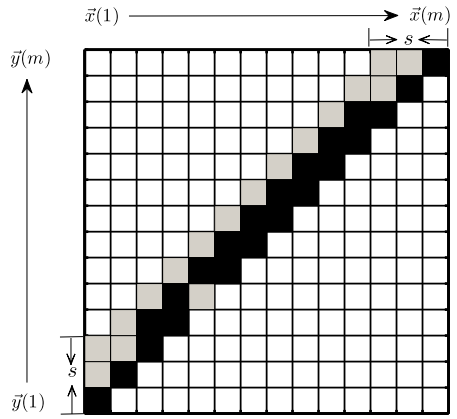


Fig. 7. DTW with a warping scope constraint $s = 2$.

Accordingly, we define the total cost as

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

where $(i_l, j_l) = p_l$ is the lth element of warping path p. Then, we can find an optimal alignment $p^* = (p_1^*, \ldots, p_L^*)$ of the minimal total cost $c_{p^*}(\vec{x}, \vec{y})$ by means of dynamic programming.

The DPC between $\vec{x}$ and $\vec{y}$, denoted as $r_{dpc}(\vec{x}, \vec{y})$, is thus given by

$$r_{dpc}(\vec{x}, \vec{y}) := \frac{1}{L-1} c_{p^*}(\vec{x}, \vec{y}), \tag{7}$$

where $L = |p^*|$. It is easy to see that $r_{dpc}(\vec{x}, \vec{y}) \leq r(\vec{x}, \vec{y}), \forall\ \vec{x}, \vec{y}$, since the Pearson correlation is a special case of DPC. Additionally, it is clear that $r_{dpc}(\vec{x}, \vec{y}) \neq r_{dpc}(\vec{y}, \vec{x})$ due to the different optimal paths, indicating that $r_{dpc}(\vec{x}, \vec{y})$ is asymmetric.

Remark. The definition of r_{dpc} in Equation (7), though logically clear, suffers from the degeneration problem, by which we mean a relatively small portion of one sequence maps onto a relatively large portion of another due to the global optimization of DTW. The warping scope s, i.e., the area that a warping path is allowed to visit in a warping matrix [Sakoe and Chiba 1978], is thus employed to avoid degeneration. That is, we restrict p_l such that $0 \leq j_l - i_l \leq s,\ \forall\ l \in [1, L]$. As illustrated in Figure 7, when $s = 2$, the black and gray areas denote all the available warping elements between $\vec{x}$ and $\vec{y}$ under this constraint, and the black areas denote the optimal warping path.

Algorithm 1 gives the procedure for computing r_{dpc}. After the initialization in Lines 1 through 8, Lines 9 through 14 compute the optimal warping value. Line 15 obtains the size of the optimal warping path, and Line 16 computes the value of r_{dpc}. Along this line, we can exhaustively compute r_{dpc} for each pair of partners in two directions and then build the PCG. Figure 5 in Section 2.1 shows an example of this process. Note that given the threshold η, only the edges with strong negative correlations are kept in the resultant PCG.

4 PARTNER PARTITIONING FOR EDGE REMOVAL

In this section, we aim to remove noise edges such that a node in a PCG has only in-edges or out-edges. To that end, we propose a graph-cut method to transform the original PCG into a bipartite graph. Table 1 gives the notations of math symbols to used below.

Table 1. Math Notations

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

ALGORITHM 1: Computing DPC

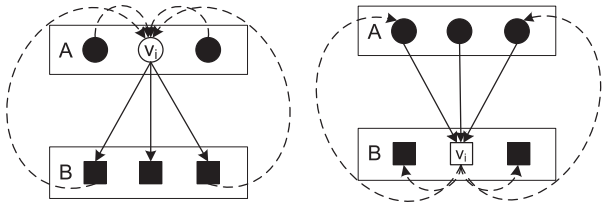
Require:
two sequences: $\vec{x}$ and $\vec{y}$
the user-specified warping scope s
Ensure:
$r_{dpc}(\vec{x}, \vec{y})$
$m := |x|$
$dtw() := new[m \times m]$
for $i := 0; i < m; i++$ **do**
  for $j := 0; j < m; j++$ **do**
    $dtw(i, j) := \infty$
  end for
end for
$dtw(0, 0) := 0$
for $i := 1; i < m; i++$ **do**
  for $j := i; j < m$ and $j \leq i + s; j++$ **do**
    $c := \frac{\vec{x}(i)-\overline{x}}{\delta_{\vec{x}}} \cdot \frac{\vec{y}(j)-\overline{y}}{\delta_{\vec{y}}}$
    $dtw(i, j) := c + \min\{dtw(i-1, j)$
    $dtw(i, j-1), dtw(i-1, j-1)\}$
  end for
end for
$L := |p^*|$ // length of the optimal warping path p^*
$r_{dpc}(\vec{x}, \vec{y}) := \frac{1}{L-1} \times dtw(m-1, m-1)$
return $r_{dpc}(\vec{x}, \vec{y})$

4.1 Greedy Algorithm for Edge Removal

Measure for the Cheating Degree. Before building the greedy algorithm, we first propose the measure Δw_i to check the degree to which node v_i is a cheating partner, either as seller or buyer,

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

where the notations used are summarized in Table 1. Note that in Equation (8), we assume that $\{A, B\}$, i.e., the partition of the node set V is given. Intuitively, as shown in Figure 8, the larger the value of Δw_i^A (Δw_i^B) is, the more likely it is that node v_i is a cheating seller (buyer).



(a) Δw_i^A (b) Δw_i^B

Fig. 8. Illustration of the cheating degree for node v_i.

Greedy Method. We now try to partition the PCG based on the cheating degree measure Δw_i. Since in reality A and B are both unknown in advance, we develop a greedy method, as shown in Algorithm 2.

In the algorithm, A' and B' denote the sets of cheating sellers and cheating buyers, respectively, in the current round. Initially, A' and B' are both empty sets. Then, in each round, we select the node $v_i \notin A' \bigcup B'$ with the maximum Δw_i value. Recall that we assume $\{A, B\}$ is known in advance for the definition of the cheating degree in Equation (8), which fails, since only $\{A', B'\}$ ($A' \subseteq A$, $B' \subseteq B$) is given in each round. Hence, we modify the measure Δw_i slightly to $\Delta w_i'$, as follows:

$$\Delta w_i' = \begin{cases} \Delta w_i^{A'} = (w_{i*} - w_{iA'}) - w_{*i}, & \text{if } v_i \in A', \\ \Delta w_i^{B'} = (w_{*i} - w_{B'i}) - w_{i*}, & \text{if } v_i \in B'. \end{cases} \tag{9}$$

Compared with the definition in Equation (8), w_{iB} and w_{Ai} are substituted with $(w_{i*} - w_{iA'})$ and $(w_{*i} - w_{B'i})$, respectively. This change is based on the fact that with the current sets of A', B', for any A, B such that $A \cap B' = \emptyset$, $B \cap A' = \emptyset$, we have

$$w_{iB} \leq (w_{i*} - w_{iA'}), \tag{10}$$

$$w_{Ai} \leq (w_{*i} - w_{B'i}). \tag{11}$$

Therefore, $(w_{i*} - w_{iA'})$ and $(w_{*i} - w_{B'i})$ are actually the upper bounds of w_{iB} and w_{Ai}, respectively. With increasing iterations, the upper bounds gradually tend toward the true values.

Based on the definition of $\Delta w_i'$, Lines 3–7 in Algorithm 2 compute $\Delta w_i'$ for all the nodes, and Line 8 chooses the node that maximizes $\Delta w_i'$. Here since we cannot determine whether node v_i is a cheating seller or buyer, we set $\Delta w_i'$ to the maximum of $\Delta w_i^{A'}$ and $\Delta w_i^{B'}$.

After the v_k with the maximum value is identified, we insert v_k into A' (or B') if $\Delta w_k^{A'} \geq \Delta w_k^{B'}$ (or $\Delta w_k^{A'} < \Delta w_k^{B'}$). Consider the situation that v_k is inserted into A'. Then, to ensure that Δw_k in the final bipartite graph is equal to $\Delta w_k^{A'}$, we consider all the nodes connected by the out-edges of v_k. All these nodes, except the ones already inside A', are placed into B', as shown in Lines 9 through 12 in Algorithm 2.

The iteration terminates when we obtain the final partition of A and B. In this bipartite graph, only the edges from A to B are reserved.

Running Example. We use the example in Figure 5 to describe the running process of the algorithm.

• Round 1. Initially, with $A' = \emptyset, B' = \emptyset$, $\Delta w_1^{A'}$ reaches the maximum value. Thus, we place node v_1 into A'. Meanwhile, to make Δw_1 in the final partition equal to $\Delta w_1^{A'}$, we also place all the nodes on the out-edges of v_1, namely, nodes v_2 and v_4, into B'. This step is conducted by Lines 9 through 16 in Algorithm 2. After this round, $A' = \{v_1\}, B' = \{v_2, v_4\}$.

• Round 2. Next, we compute the values of $\Delta w_3^{A'}, \Delta w_3^{B'}, \Delta w_5^{A'}$, and $\Delta w_5^{B'}$. The sub-figures in Figure 9 illustrate the process of computing $\Delta w_3^{A'}$ and $\Delta w_3^{B'}$.

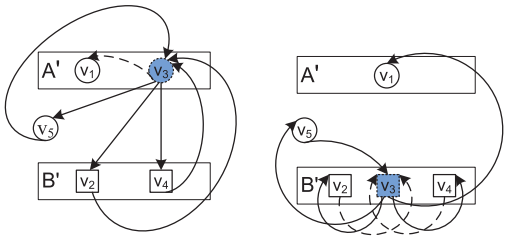


Fig. 9. Illustration of computing $\Delta w_3'$ with the greedy algorithm (only the edges of v_3 are shown).

ALGORITHM 2: Greedy Algorithm for Edge Removal

Require:
a directed graph $G = (V, E, w)$
Ensure:
a bipartite graph with $E' := \{(v_i, v_j) \in E$
$\mid v_i \in A$ and $v_j \in B\}$
$n := |V|, A', B' := \{\}$
/* A' and B' denote current set of A and B */
while $A' \cup B' \neq V$ **do**
  for all $i \notin A'$ and $i \notin B'$ **do**
    $\Delta w_i^{A'} := (w_{i*} - w_{iA'}) - w_{*i}$
    $\Delta w_i^{B'} := (w_{*i} - w_{B'i}) - w_{i*}$
    $\Delta w_i' := \max(\Delta w_i^{A'}, \Delta w_i^{B'})$
  end for
  $k := \arg\max_i \Delta w_i'$
  if $\Delta w_k' = \Delta w_k^{A'}$ **then**
    put v_k **into** A'
    put all v_j **into** B' **if** $(v_k, v_j) \in E_{k*}$ **and** $v_j \notin A'$
    // E_{k*} denotes all out-edges of node v_k
  end if
  if $\Delta w_k' = \Delta w_k^{B'}$ **then**
    put v_k **into** B'
    put all v_j **into** A' **if** $(v_j, v_k) \in E_{*k}$ **and** $v_j \notin B'$
    // E_{*k} denotes all in-edges of node v_k
  end if
end while
return A', B'

As shown in Figure 9(a), we compute $\Delta w_3^{A'}$ if v_3 is placed into A'. Here, v_3 has four out-edges, but the weight on edge (v_3, v_1) cannot be considered for $v_1 \in A'$. Thus, we have

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

Similarly, as shown in Figure 9(b), we compute $\Delta w_3^{B'}$ if v_3 is placed into B'. Here, v_3 has three in-edges, but the weights (v_2, v_3) and (v_4, v_3) must also be omitted for $v_2 \in B', v_4 \in B'$. Thus, we have

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

Along this line, we obtain $\Delta w_5^{A'} = -0.75$ and $\Delta w_5^{B'} = -0.7$. Clearly, $\Delta w_3^{B'}$ is the maximum value among $\Delta w_3^{A'}$, $\Delta w_3^{B'}$, $\Delta w_5^{A'}$, and $\Delta w_5^{B'}$; therefore, we place node v_3 into B'. Then, we consider all the nodes on the in-edges of node v_3, namely, v_2, v_4, and v_5. Since v_2 and v_4 are already in B', we place only v_5 into A', which gives the final partition result as follows:

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

4.2 Graph Cut: General Problem and Solution

The problem of finding the partition (A, B) for a directed weighted graph V such that the specified objective function of the partition is maximized is actually the well-known *graph-cut* problem. The greedy method proposed in Section 4.1 can be viewed as an ad hoc solution to this problem. In this section, we reformulate the problem with a new objective function based on the application scenario of our task and propose a general SDP solution. We also compare the performance with that of the greedy algorithm via experiments.

4.2.1 Problem Formulation for Graph Cut: Max-DifCut. Most of the previous studies in this area focus on the problem of *Max-DiCut* [Matuura and Matsui 2001], 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 study, 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.

Recall the definition of cheating degree in Equation (8). With this measure, we aim to find the partition that maximizes the measure for each node. Therefore, by summing the measure across all the nodes, we obtain the following function:

$$\begin{aligned}
&\sum_{v_i \in A} (w_{iB} - w_{*i}) + \sum_{v_j \in B} (w_{Aj} - w_{j*}) \\
&= (w_{AB} - w_{*A}) + (w_{AB} - w_{B*}) \\
&= (w_{AB} - (w_{BA} + w_{AA})) + (w_{AB} - (w_{BA} + w_{BB})) \\
&= 3w_{AB} - w_{BA} - w_G. \qquad (12)
\end{aligned}$$

Note that w_G in Equation (12) is a constant for any graph G. Therefore, the new graph-cut problem is to find a partition (A, B) of E such that $3w_{AB} - w_{BA}$ is maximized. Clearly, the method proposed in Section 4.1 is a greedy solution to this problem, since in each round, the method identifies the node that can increase the objective function the most.

4.2.2 The Semi-Definite Programming Solution. Since semi-definite programming is regarded as the state-of-the-art solution for graph-cut problems [Goemans and Williamson 1995], we also present an SDP solution to Max-DifCut and discuss the bounds of the approximation ratio for the SDP relaxation. We theoretically prove that the SDP method returns a solution whose objective value is at least 0.531 times the true optimal solution. The details are given as follows.

SDP Solution to Max-DifCut. For a node v_i, let t_i be equal to 1 if v_i belongs to A and 0 otherwise, and let f_i be equal to 1 if v_i does not belong to A and 0 otherwise. As a result, we have $t_i + f_i = 1$ and $t_i f_i = 0$. Although t_i and f_i are distinct 0-1 values, we can obtain an upper bound on the optimum value by relaxing all the t_i and f_i to be n-dimensional vectors, i.e., $\vec{t_i}$ and $\vec{f_i}$, instead

of scalars. Thus, we obtain the following form of the Max-DifCut problem:

$$\begin{aligned} \text{Maximize} \quad & \sum_{(v_i,v_j)\in E} 3\, w_{ij}(\vec{t}_i \cdot \vec{f}_j) - w_{ij}(\vec{f}_i \cdot \vec{t}_j) \\ \text{subject to:} \quad & \vec{t}_i \cdot \vec{f}_i = 0 && \forall v_i \in V \\ & \vec{t}_i + \vec{f}_i = \vec{q}_0 && \forall v_i \in V \\ & \vec{t}_i, \vec{f}_i, \vec{q}_0 \in \mathbb{R}^n && \forall v_i \in V, \end{aligned} \tag{13}$$

where $\vec{q}_0$ is any unit vector (i.e., $\vec{q}_0 = (1, 0, \ldots, 0)^\top$). For a node v_i, we define a vector $\vec{q}_i$ such that $\vec{t}_i = \frac{1}{2}(\vec{q}_0 + \vec{q}_i)$ and $\vec{f}_i = \frac{1}{2}(\vec{q}_0 - \vec{q}_i)$ and that satisfies the conditions in Equation (13). We also add some valid constraints, such as $\vec{t}_i \cdot \vec{t}_j \geq 0$, $\vec{t}_i \cdot \vec{f}_j \geq 0$, $\vec{f}_i \cdot \vec{t}_j \geq 0$, and $\vec{f}_i \cdot \vec{f}_j \geq 0$. Finally, we obtain the following relaxed problem:

$$\begin{aligned} \text{Maximize} \quad & \sum_{(v_i,v_j)\in E} w_{ij}\left(\frac{1}{2} + \vec{q}_0 \cdot \vec{q}_i - \vec{q}_0 \cdot \vec{q}_j - \frac{1}{2}\vec{q}_i \cdot \vec{q}_j\right) \\ \text{subject to:} \quad & \vec{q}_i \cdot \vec{q}_i = 1, && \forall v_i \in V \\ & \vec{q}_i \in \mathbb{R}^n, && \forall v_i \in V \\ & \vec{q}_0 \cdot \vec{q}_i + \vec{q}_0 \cdot \vec{q}_j + \vec{q}_i \cdot \vec{q}_j \geq -1, && \forall (v_i, v_j) \in E \\ & \vec{q}_0 \cdot \vec{q}_i - \vec{q}_0 \cdot \vec{q}_j - \vec{q}_i \cdot \vec{q}_j \geq -1, && \forall (v_i, v_j) \in E \\ & -\vec{q}_0 \cdot \vec{q}_i - \vec{q}_0 \cdot \vec{q}_j + \vec{q}_i \cdot \vec{q}_j \geq -1, && \forall (v_i, v_j) \in E \\ & -\vec{q}_0 \cdot \vec{q}_i + \vec{q}_0 \cdot \vec{q}_j - \vec{q}_i \cdot \vec{q}_j \geq -1, && \forall (v_i, v_j) \in E. \end{aligned} \tag{14}$$

Then, we can solve the above problem by semi-definite programming in polynomial time and obtain the final partition by the following random rounding step. Let $(\vec{q}'_1, \vec{q}'_2, \ldots, \vec{q}'_m)$ be the optimal solution of the optimized problem in Equation (14). We generate a vector $\vec{r}$ uniformly satisfying $\vec{r} \in \mathbb{R}^n$ and $\vec{r} \cdot \vec{r} = 1$. Then, we round the SDP solution to a partition (A, B) of V as follows:

$$\begin{cases} A = \{v_i : sign(\vec{r} \cdot \vec{q}_0) = sign(\vec{r} \cdot \vec{q}'_i)\}, \\ B = \{v_j : sign(\vec{r} \cdot \vec{q}_0) \neq sign(\vec{r} \cdot \vec{q}'_j)\}. \end{cases} \tag{15}$$

This rounding process outputs w_{AB} and w_{BA} and then returns the optimal value SOL as below,

$$SOL = \max(3\, w_{AB} - w_{BA}, 3\, w_{BA} - w_{AB}). \tag{16}$$

Upper Bound of SDP Approximation Ratio. Next, we discuss the approximation ratio of SDP to Max-DifCut. Since the rounding step is performed randomly, we denote $E(w_{AB})$ as the expected value of the directed cut from A to B for the rounding step after the SDP process for maximizing w_{AB}. Additionally, we denote $\text{SDP}(w_{AB})$ as the optimal value of the SDP relaxation problem for maximizing w_{AB} and denote $\text{SDP}(3w_{AB} - w_{BA})$ as the optimal value of the SDP relaxation problem for maximizing $3w_{AB} - w_{BA}$. We then bound the approximation ratio as follows:

$$\begin{aligned} \text{E}[SOL] &\geq 2\text{E}(w_{AB}), && (17) \\ &\geq 0.796 \cdot 2\text{SDP}(w_{AB}), && (18) \\ &\geq 0.796 \cdot \text{SDP}\left(2w_{AB} - \frac{2}{3}w_{BA}\right), && (19) \\ &= 0.796 \cdot \frac{2}{3}\text{SDP}(3w_{AB} - w_{BA}), && (20) \\ &\geq 0.531\ \text{SDP}(3w_{AB} - w_{BA}), && (21) \\ &\geq 0.531\ \max_{A,B}(3w_{AB} - w_{BA}). && (22) \end{aligned}$$

Note that Equation (17) follows from the definition of SOL in Equation (16). Equation (18) is guaranteed by the approximation ratio of the Max Directed-Cut SDP algorithm [Goemans and Williamson 1995]. Equation (22) comes from the SDP relaxation.

Remark. Intuitively, the SDP solution should be better than the greedy solution in terms of the objective function. Therefore, from the perspective of optimizing the objective function of Max-DifCut, the SDP solution can achieve a better solution than that of the greedy method. However, in the experimental section to follow, we show that the greedy solution could be better in terms of the final ranking of cheating partners (see Tables 4 to 6 below), although the SDP solution indeed shows the better results of graph cut (see Figure 11 below).

To understand this apparent contradiction, let us revisit Lines 11 and 15 in Algorithm 2, which place the directed neighborhood nodes of the selected node into the opposite set. These steps consolidate the role of the selected node, either as a cheating seller or cheating buyer, who has a high probability of entering the final ranking list as a top-k cheating partner. The directed neighborhood nodes, which might not play the contrary role of the selected node according to the ground truth, have much less impact on the ranking result, because they are usually not in the top-k list. In other words, the greedy method does well in recognizing the top-k cheating partners while the SDP method achieves this recognition in a more global manner. Nevertheless, the SDP solution could be particularly useful in applications where optimizing the objective function of Max-DifCut in Equation (12) is the direct goal. More details of the empirical comparison of these two methods are given in the experimental section.

5 PARTNERS RANKING

Here, we propose methods for partners ranking in the following three scenarios.

Ranking Based on Resultant Bipartite Graph. After the graph-cut step, we obtain a bipartite graph containing only the edges from the cheating-seller side A to the cheating-buyer side B. Then, for any node v_i, the ranking score is defined as

$$score(i) = \begin{cases} \sum_{j \in B} w_{ij}, & \text{if } v_i \in A, \\ \sum_{j \in A} w_{ji}, & \text{if } v_i \in B. \end{cases} \tag{23}$$

We call this ranking method *Cut_Rank*, because it uses the graph-cut method in advance. By default, the greedy graph-cut method is adopted here. If the SDP method is used instead, then the corresponding ranking method is denoted as *Cut′_Rank*.

Ranking Based on the Original Directed Graph. To deliberately show that the graph-cut step improves the ranking performance, we also develop the following ranking score for comparison, which is based on the original directed graph PCG. Specifically, for any node v_i in PCG, the ranking score is given by

$$score(i) = |w_{i*} - w_{*i}|, \tag{24}$$

where w_{i*} and w_{*i} are the weight sums of the out-edges and in-edges, respectively. This score considers the difference between these two values. We call this ranking method *Noncut_Rank*, since no graph-cut method is used in advance.

Ranking Based on Undirected Graph. Moreover, to deliberately show that the proposed asymmetric correlation measure improves the ranking performance, we propose a ranking method based on the symmetric Pearson correlation. The computation is as follows. For each pair of partners, we first compute the symmetric Pearson correlation between the two corresponding sequences of sales volumes and then generate an undirected graph of partner correlations, where each node represents an individual partner and the edge weight is set to the negative value of the Pearson correlation between the two linked nodes. A user-specified parameter $0 < \eta < 1$ is also

Table 2. Summary of the Ranking Methods for Comparison

Name	Description
PearsonRank	undirected graph used
Noncut_Rank	directed graph used, without using the graph-cut method
Cut_Rank	bipartite graph used by employing the greedy graph-cut method on the directed graph
Cut′_Rank	bipartite graph used by employing the SDP graph-cut method on the directed graph

used to remove the edges with weights less than η. The ranking score of a node is finally defined as the weight sum of its linked edges. We call this method *PearsonRank*.

In Table 2, we summarize the three ranking methods, which are subjected to validity checking and comparison based on a real-world dataset in the experimental section.

6 EXPERIMENTAL EVALUATION

In this section, we present the experimental results on a real-world dataset obtained from a worldwide IT company.

6.1 Experimental Datasets

We first introduce the dataset used in this section for experimental evaluation. This dataset was obtained from the market channel of a worldwide IT company. All the partners are divided into two types, namely, *gold membership partners* and *silver membership partners*. There are a total of 104 gold membership partners and 424 silver membership partners. The sequence of the monthly purchase volume is given for each partner. The time interval of the sequence is from January 2009 to December 2012, a total of 48 months. By means of this dataset, experiments are conducted on two groups of partners, namely, gold membership partners (*Gold* for short) and all partners (*All* for short). Moreover, we have a blacklist of real cheating partners as the ground-truth data. There are 17 and 85 true cheating partners among the gold and silver membership partners, respectively. However, their roles as either cheating sellers or cheating buyers are unclear. We therefore aim to rank the real cheating partners as high as possible via the proposed method without considering their roles.

6.2 Evaluation Measures and Model Parameters

With the blacklist of true cheating partners, we adopt the following measures to evaluate the proposed method. Assume we have M true cheating partners. We then count the number of hits on M in the top-k ranking list of cheating partners and define *precision*, *recall*, and $F1$ as follows:

$$\begin{aligned} precision@k &= \frac{\text{Number of hits in the top-}k\text{ list}}{k}, \\ recall@k &= \frac{\text{Number of hits in the top-}k\text{ list}}{M}, \\ F1@k &= 2 \cdot \frac{precision@k \times recall@k}{precision@k + recall@k}. \end{aligned} \tag{25}$$

The above measures are clearly sensitive to k. To address this issue, we plot the curves, as shown in Figure 10, where the X axis represents k and the Y axis shows the measure values. Then, we calculate the area under the curve (AUC) for precision, recall and F1 and denote these values

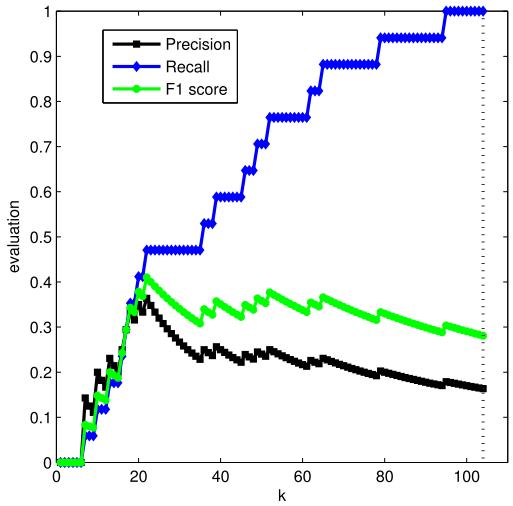


Fig. 10. Example curves of precision, recall and F1 score.

Table 3. Parameter Settings

Real-world 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}

as *AUC_p*, *AUC_r* and *AUC_F*1, respectively. A higher AUC indicates better performance of a ranking method.

The compared ranking methods are summarized in Table 2. For *Noncut_Rank*, *Cut_Rank* and *Cut′_Rank*, we have two parameters: the dynamic warping scope s and the threshold η for PCG edge removal. The ranges of these parameters are set as follows:

- DTW scope s. According to a market investigation, a partner typically stores products for at most three months. Thus, we select s from $\{1, 2, 3\}$.
- Threshold η. η is set to remove edges whose negative correlations are not sufficiently large to indicate cheating behavior. For an over-large η, however, the resultant PCG may contain many isolated nodes that cannot be ranked, because they do not have any in-edges and out-edges. Thus, we set the range of η carefully so that the generated PCG remains a fully connected graph.

Table 3 shows the range of η under different configurations of s for the real-world dataset. The parameter for the *PearsonRank* method (only η) is set in a similar way.

6.3 Experimental Results

Here, we show the results for two real-world datasets. For each setting of s and η in Table 3, we compute *AUC_p*, *AUC_r*, and *AUC_F*1 for each method. The best and the average values over all the parameter settings for each method are reported in Tables 4 and 5.

Table 4. The Best Performance of Each Method on Real-World Datasets

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
	Cut_Rank	27.40	70.91	**35.22**
	Cut′_Rank	29.20	66.59	33.91
All	*PearsonRank*	101.00	308.36	139.70
	Noncut_Rank	82.00	259.64	111.64
	Cut_Rank	120.68	345.45	**162.27**
	Cut′_Rank	128.01	324.84	157.80

Table 5. The Average Performance of Each Method on Real-World Datasets

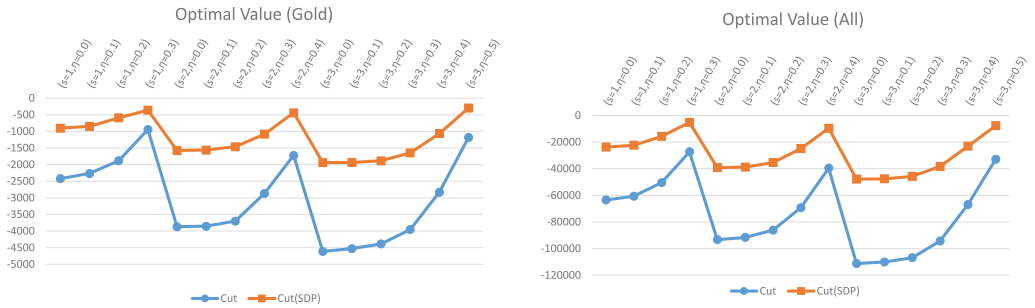
Dataset	Method	AUC		
		AUC_p	*AUC_r*	*AUC_F1*
Gold	*PearsonRank*	19.80	56.22	25.82
	Noncut_Rank	15.01	50.05	21.68
	Cut_Rank	23.94	64.62	**31.35**
	Cut′_Rank	24.81	61.70	30.49
All	*PearsonRank*	99.48	303.86	137.50
	Noncut_Rank	73.47	246.79	104.26
	Cut_Rank	110.17	372.87	**150.48**
	Cut′_Rank	115.56	314.50	148.18

Table 6. The *t*-Test on Each Pair of Methods on Real-World Datasets

Dataset	Pairs of Methods	*p*-value
Gold	(*Noncut_Rank*, *Cut_Rank*)	1.57×10^{-7}
	(*Noncut_Rank*, *Cut′_Rank*)	5.65×10^{-8}
	(*Cut_Rank*, *Cut′_Rank*)	0.185
All	(*Noncut_Rank*, *Cut_Rank*)	3.43×10^{-11}
	(*Noncut_Rank*, *Cut′_Rank*)	1.72×10^{-11}
	(*Cut_Rank*, *Cut′_Rank*)	0.015

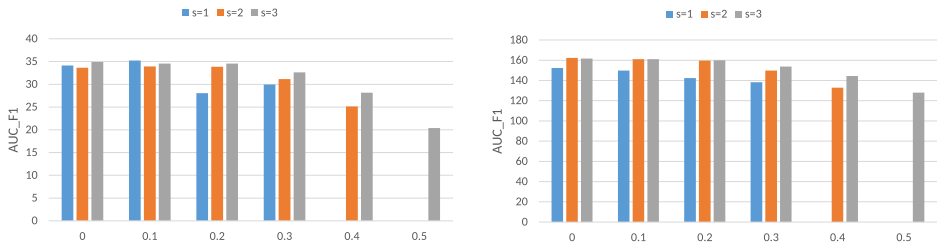
Moreover, for each pair of ranking methods over the directed graph, we also perform the *t*-test to check whether the difference in performance of the two corresponding methods is statistically significant. The *p*-values are shown in Table 6. The three tables present the following findings:

- *Cut_Rank* and *Cut′_Rank* are much better than *Nocut_Rank* and *PearsonRank* in terms of both the best and average performance over all the model parameters. Therefore, the asymmetric correlation measure along with the graph-cut method can detect cheating behavior.
- *PearsonRank* is slightly better than *Nocut_Rank*, which indicates that the asymmetric correlation measure *alone* is not responsible for the performance improvement. The measure



(a) On Dataset *Gold* (b) On Dataset *All*

Fig. 11. Optimal value of Max-DifCut: Comparison of the greedy and SDP solutions.



(a) On Dataset *Gold* (b) On Dataset *All*

Fig. 12. Analysis of the sensitivity of s and η to *Cut_Rank*.

introduces more information into the system while also introducing some noise. This illustrates the critical importance of the successive graph cut on PCG.

- Although the t-test shows no significant difference, *Cut_Rank* is better than *Cut′_Rank* in terms of both the best and average performance. This is consistent with our discussion at the end of Section 4.2.2; that is, while *Cut′_Rank* (the SDP solution) can achieve a better graph cut (shown in Figure 11), *Cut_Rank* (the greedy solution) achieves better performance in the final cheating detection.

Moreover, we study how the parameters s and η affect the ranking performance of *Cut_Rank*. As shown in Figure 12, an increase in s generally improves the ranking performance, which indicates that cheating behavior may occur with a three-month time delay. Additionally, Figure 12 shows that a smaller η results in better ranking performance. This somewhat unexpected result implies that edges with relatively small weights are also valuable for the final cheating detection.

7 RELATED WORK

In this section, we briefly review the works related to our study, which can be divided into three categories as follows.

Pearson correlation. The Pearson correlation [Lee Rodgers and Nicewander 1988] is a well-known measure of the linear correlation of two dependent variables and has been widely used in different domains, such as recommendation [Adomavicius and Zhang 2016] and ranking [Zhang et al. 2016]. This method was first developed by Karl Pearson and is widely used to measure the similarity between pairwise time series. Warren Liao [2005] used the measure to cluster time series. Papadimitriou et al. [2006] proposed a local correlation measure to track the correlation among

time-evolving time series. Xiong et al. [2004] explored the upper bound of the Pearson correlation to identify strongly correlated pair-wise time sequences. Kawale et al. [2012] used the correlation measure to discover dipoles, which represent long distance connections between the pressure anomalies of two distant regions that are negatively correlated. They proposed a method to evaluate the significance of the correlations. We also applied 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. One important aspect is to measure the correlations among nodes by comparing the time-series patterns. With the recent advancements of deep neural networks such as recurrent neural networks, we can learn the representations of time series by neural temporal point process modeling, and further compute their correlations. For example, Du et al. proposed a recurrent temporal point process framework to model the event timings and markers simultaneously through a LSTM network structure [Du et al. 2016]. Based on that, Xiao et al. proposed an end-to-end framework modeling the intensity function of temporal point process by using RNNs for both background and history effects [Xiao et al. n.d.]. However, arbitrarily utilizing deep learning for correlation computation of time series may face additional challenges. For example, the interpretability of the learned representation is not clear, and the warping range of time-series pairs cannot be controlled and adjusted. In this article, the proposed dynamic Person correlation can explicitly use warping range to guide the correlation computation for time series for measuring cheating degrees.

Dynamic Time Warping. DTW is widely used in various fields such as bioinformatics, chemical engineering and robotics [Aach and Church 2001; Giannoula et al. 2018; Kovacs Vajna 2000; Munich and Perona 1999]. Keogh and Ratanamahatana [2005] applied DTW to exactly index time series from a large database by proposing a lower bound of the DTW distance. In Keogh and Pazzani [2000, 2001], DTW was modified to approximate a high level abstraction of data and to track the local accelerations and decelerations on the time axis. Other studies developed the concept of correlation-optimized warping for chromatographic data analysis [Tomasi et al. 2004; Zhang et al. 2008]. They conducted segmentwise correlation optimization, in which the sequence is segmented into pieces and the warping is applied at the segment level. Recently, Tan et al. [2018] explored a principled way to decide the best warping window for DTW. Zhao and Itti [2018] proposed an efficient DTW algorithm to model local shape for time-series classification. However, our measure directly applies warping at the point level. Most importantly, to the best of our knowledge, our article is the first study on leveraging DTW to model the time series for cheating detection.

Graph cut methods. Most of the studies in this area, however, focus on the Max-DiCut problem with the objective function of w_{AB} [Goemans and Williamson 1994, 1995; Halperin and Zwick 2001; Matuura and Matsui 2001]. The graph partitioning problem is a representative combinatorial optimization problem that is NP-hard [Buluc et al. 2013]. Existing research mainly focuses on approximate, efficient, heuristic, and evolutionary algorithms to solve graph partitioning problems [Kim and Kim 2018; Orecchia 2011]. Heuristic algorithms aim to solve graph partitioning optimally [Barucca 2017; Björklund et al. 2009]. Orecchia [2011] proposed a fast approximate algorithm to optimize the graph cut objective with spectral and SDP programming. Kim and Kim [2018] gave an overview of recent advanced evolutionary methods for graph partitioning. n this study, motivated by the application background of cheating detection we propose a new graph-cut problem, called Max-DifCut, with the objective function of $3w_{AB} - w_{BA}$ (detailed in Equation (12)). We propose both greedy and approximate solutions using SDP for the proposed graph cut problem, with comparison of the optimal guarantee and ranking performance for cheating detection. Currently, we learn node ranking score based on derived node weights from the bipartite

graph. Inspired by Liu et al. [2017], we could also incorporate the prior knowledge such as the profile/attributes of each partner, as the constraints for better node ranking.

8 CONCLUSION

In this article, we leverage correlation network analysis for cheating detection in distribution channels. Our solution consists of three modules: (1) building correlation networks via computing the novel directed Pearson correlation to generate the partner correlation graph; (2) partitioning all partners in the PCG into suspicious cheating sellers and buyers via greedy and dynamic programming methods; and (3) ranking the partners based on the resultant bipartite graph. The experiments on real-world datasets show that the method *Cut_Rank*, containing all three modules, performs the best in detecting cheating partners. In general, we develop a framework to rank the nodes inside the system according to a specified ranking measure. In this sense, this method could help to open a wide application area of correlation network analysis and motivate more interesting applications. For future work, we could (1) exploit advanced deep learning models for modeling time series for correlation embedding and (2) incorporate the prior knowledge in our specific domain such as the profile/attributes of each partner, as the constraints for node ranking.

ACKNOWLEDGMENTS

A preliminary version of this work has been published in the *Proceedings of the 17th ACM Conference on Information and Knowledge Management* (CIKM) [Shu et al. 2014]. The authors are grateful to the anonymous referees for their constructive comments on this article.

REFERENCES

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

Gediminas Adomavicius and Jingjing Zhang. 2016. Classification, ranking, and top-K stability of recommendation algorithms. *INFORMS J. Comput.* 28, 1 (2016), 129–147.

Ziv Bar-Joseph. 2004. Analyzing time series gene expression data. *Bioinformatics* 20, 16 (2004), 2493–2503.

Paolo Barucca. 2017. Spectral partitioning in equitable graphs. *Phys. Rev. E* 95, 6 (2017), 062310.

Yehiel Berezin, Avi Gozolchiani, O. Guez, and S. Havlin. 2012. Stability of climate networks with time. *Sci. Rep.* 2 (2012), 666.

Andreas Björklund, Thore Husfeldt, and Mikko Koivisto. 2009. Set partitioning via inclusion-exclusion. *SIAM J. Comput.* 39, 2 (2009), 546–563.

Giovanni Bonanno, Guido Caldarelli, Fabrizio Lillo, Salvatore Micciché, Nicolas Vandewalle, and Rosario Nunzio Mantegna. 2004. Networks of equities in financial markets. *Eur. Phys. J. B-Condens. Matter Complex Syst.* 38, 2 (2004), 363–371.

Aydın Buluç, Henning Meyerhenke, Ilya Safro, Peter Sanders, and Christian Schulz. 2016. Recent Advances in Graph Partitioning. In *Algorithm Engineering*. Springer, Cham, 117–158.

Journal F. Donges, Y. Zou, N. Marwan, and Journal Kurths. 2009. Complex networks in climate dynamics—Comparing linear and nonlinear network construction methods. *Eur. Phys. J. Spec. Top.* 174 (2009), 157–179.

Nan Du, Hanjun Dai, Rakshit Trivedi, Utkarsh Upadhyay, Manuel Gomez-Rodriguez, and Le Song. 2016. Recurrent marked temporal point processes: Embedding event history to vector. In *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD'16)*.

Alexia Giannoula, Alba Gutierrez-Sacristán, Álex Bravo, Ferran Sanz, and Laura I. Furlong. 2018. Identifying temporal patterns in patient disease trajectories using dynamic time warping: A population-based study. *Sci. Rep.* 8, 1 (2018), 4216.

Michel X. Goemans and David P. Williamson. 1994. 0.879-approximation algorithms for MAX CUT and MAX 2SAT. In *Proceedings of the 26th Annual ACM Symposium on Theory of Computing (STOC'94)*. 422–431.

Michel X. Goemans and David P. Williamson. 1995. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *J. ACM* 42, 6 (1995), 1115–1145.

Eran Halperin and Uri Zwick. 2001. Combinatorial approximation algorithms for the maximum directed cut problem. In *Proceedings of the 12th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'01)*. 1–7.

Kenneth Hardy and Allan Magrath. 1989. Dealing with cheating in distribution. *Eur. J. Market.* 23, 2 (1989), 123–129.

Jaya Kawale, Snigdhansu Chatterjee, Dominick Ormsby, Karsten Steinhaeuser, Stefan Liess, and Vipin Kumar. 2012. Testing the significance of spatio-temporal teleconnection patterns. In *Proceedings of the 18th ACM SIGKDD International Conference Knowledge Discovery and Data Mining (KDD'12)*. 642–650.

Dror Y. Kenett, Yoash Shapira, Asaf Madi, Sharron Bransburg-Zabary, Gitit Gur-Gershgoren, and Eshel Ben-Jacob. 2010. Dynamics of stock market correlations. *AUCO Czech Econ. Rev.* 4, 3 (2010), 330–341.

Eamonn Keogh and Chotirat Ann Ratanamahatana. 2005. Exact indexing of dynamic time warping. *Knowl. Inf. Syst.* 7, 3 (2005), 358–386.

Eamonn J. Keogh and Michael J. Pazzani. 2000. Scaling up dynamic time warping for data mining applications. In *Proceedings of the 16th ACM SIGKDD International Conference Knowledge Discovery and Data Mining (KDD'00)*. 285–289.

Eamonn J. Keogh and Michael J. Pazzani. 2001. Derivative dynamic time warping. In *Proceedings of the 1st SIAM International Conference Data Mining (SDM'01)*. 5–7.

Hye-Jin Kim and Yong-Hyuk Kim. 2018. Recent progress on graph partitioning problems using evolutionary computation. *arXiv:1805.01623* (2018).

Zsolt Miklos Kovacs Vajna. 2000. A fingerprint verification system based on triangular matching and dynamic time warping. *IEEE Trans. Pattern Anal. Mach. Intell.* 22, 11 (2000), 1266–1276.

Joseph Lee Rodgers and W. Alan Nicewander. 1988. Thirteen ways to look at the correlation coefficient. *Am. Stat.* 42, 1 (1988), 59–66.

Qi Liu, Biao Xiang, Nicholas Jing Yuan, Enhong Chen, Hui Xiong, Yi Zheng, and Yu Yang. 2017. An influence propagation view of pagerank. *ACM Trans. Knowl. Discov. Data* 11, 3 (2017), 30.

Josef Ludescher, Avi Gozolchiani, Mikhail I. Bogachev, Armin Bunde, Shlomo Havlin, and Hans Joachim Schellnhuber. 2014. Very early warning of next El Niño. *Proc. Natl. Acad. Sci. U.S.A.* 111, 6 (2014), 2064–2066.

Rosario N. Mantegna. 1999. Hierarchical structure in financial markets. *Eur. Phys. J. B-Condens. Matter Complex Syst.* 11, 1 (1999), 193–197.

Shiro Matuura and Tomomi Matsui. 2001. 0.863-approximation algorithm for MAX DICUT. In *Proceedings of the 4th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX'01)*. 138–146.

Mario E. Munich and Pietro Perona. 1999. Continuous dynamic time warping for translation-invariant curve alignment with applications to signature verification. In *Proceedings of the 7th IEEE International Conference Computer Vision (ICCV'99)*. 108–115.

Das Narayandas and V. Kasturi Rangan. 2004. Building and sustaining buyer-seller relationships in mature industrial markets. *J. Market.* 68, 3 (2004), 63–77.

Lorenzo Orecchia. 2011. *Fast Approximation Algorithms for Graph Partitioning Using Spectral and Semidefinite-programming Techniques*. Ph.D. Dissertation. UC Berkeley.

Spiros Papadimitriou, Jimeng Sun, and Philip S. Yu. 2006. Local correlation tracking in time series. In *Proceedings of the 6th IEEE International Conference Data Mining (ICDM'06)*. 456–465.

Hiroaki Sakoe and Seibi Chiba. 1978. Dynamic programming algorithm optimization for spoken word recognition. *Acoust. Speech Sign. Process.* 26, 1 (1978), 43–49.

Kai Shu, Ping Luo, Wan Li, Peifeng Yin, and Linpeng Tang. 2014. Deal or deceit: Detecting cheating in distribution channels. In *Proceedings of the 23rd ACM International Conference Information and Knowledge Management (CIKM'14)*. 1419–1428.

Ralf Steuer, Jürgen Kurths, Oliver Fiehn, and Wolfram Weckwerth. 2003. Observing and interpreting correlations in metabolomic networks. *Bioinformatics* 19, 8 (2003), 1019–1026.

Chang Wei Tan, Matthieu Herrmann, Germain Forestier, Geoffrey I. Webb, and François Petitjean. 2018. Efficient search of the best warping window for dynamic time warping. In *Proceedings of the 2018 SIAM International Conference on Data Mining*. SIAM, 225–233.

Giorgio Tomasi, Frans van den Berg, and Claus Andersson. 2004. Correlation optimized warping and dynamic time warping as preprocessing methods for chromatographic data. *J. Chemometr.* 18, 5 (2004), 231–241.

Michele Tumminello, Fabrizio Lillo, and Rosario N. Mantegna. 2010. Correlation, hierarchies, and networks in financial markets. *J. Econ. Behav. Org.* 75, 1 (2010), 40–58.

T. K. Vintsyuk. 1968. Speech discrimination by dynamic programming. *Cybernet. Syst. Anal.* 4, 1 (1968), 52–57.

T. Warren Liao. 2005. Clustering of time series data: A survey. *Pattern Recogn.* 38, 11 (2005), 1857–1874.

Shuai Xiao, Junchi Yan, Xiaokang Yang, Hongyuan Zha, and Stephen M. Chu. 2017. Modeling the Intensity Function of Point Process via Recurrent Neural Networks. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI'17)*. AAAI Press, 1597–1603.

Hui Xiong, Shashi Shekhar, Pang-Ning Tan, and Vipin Kumar. 2004. Exploiting a support-based upper bound of Pearson's correlation coefficient for efficiently identifying strongly correlated pairs. In *Proceedings of the 10th ACM SIGKDD International Conference Knowledge Discovery and Data Mining (KDD'04)*. 334–343.

D. Zhang, X. Huang, F. E. Regnier, and M. Zhang. 2008. Two-dimensional correlation optimized warping algorithm for aligning GC × GC-MS data. *Anal. Chem.* 80, 8 (2008), 2664–2671.

Zunqiang Zhang, Guoqing Chen, Jin Zhang, Xunhua Guo, and Qiang Wei. 2016. Providing consistent opinions from online reviews: A heuristic stepwise optimization approach. *INFORMS J. Comput.* 28, 2 (2016), 236–250.

Jiaping Zhao and Laurent Itti. 2018. shapeDTW: Shape dynamic time warping. *Pattern Recogn.* 74 (2018), 171–184.

Received December 2018; revised July 2019; accepted September 2019