Spectral Clustering of Customer Transaction Data With a Two-Level Subspace Weighting Method

Xiaojun Chen, *Member, IEEE*, Wenya Sun, Bo Wang, Zhihui Li, Xizhao Wang, *Fellow, IEEE*, and Yunming Ye

***Abstract*—Finding customer groups from transaction data is very important for retail and e-commerce companies. Recently, a "Purchase Tree" data structure is proposed to compress the customer transaction data and a local PurTree spectral clustering method is proposed to cluster the customer transaction data. However, in the PurTree distance, the node weights for the children nodes of a parent node are set as equal and the differences between different nodes are not distinguished. In this paper, we propose a two-level subspace weighting spectral clustering (TSW) algorithm for customer transaction data. In the new method, a PurTree subspace metric is proposed to measure the dissimilarity between two customers represented by two purchase trees, in which a set of level weights are introduced to distinguish the importance of different tree levels and a set of sparse node weights are introduced to distinguish the importance of different tree nodes in a purchase tree. TSW learns an adaptive similarity matrix from the local distances in order to better uncover the cluster structure buried in the customer transaction data. Simultaneously, it learns a set of level weights and a set of sparse node weights in the PurTree subspace distance. An iterative optimization algorithm is proposed to optimize the proposed model. We also present an efficient method to compute a regularization parameter in TSW. TSW was compared with six clustering algorithms on ten benchmark data sets and the experimental results show the superiority of the new method.**

***Index Terms*—Clustering, clustering tree, customer segmentation, two level weighting.**

I. Introduction

TRANSACTION data is the collection of daily shopping transactions of customers at a retail company. As one of the most critical tasks in modern marketing and customer relationship management, clustering of customer transaction data is used to partition customers into different customer groups based on their purchase behaviors, so that the customers in the same cluster bought more similar goods to each other than to those in other clusters. The early segmentation methods use general variables like customer demographics, lifestyle, attitude, and psychology, because such variables are intuitive and easy to operate [1]. With the rapid increase of customer behavior data, the new study turns to use product specific variables such as items purchased [2]–[4]. Although some methods which cluster item set data were proposed [4]–[6], these methods are time-consuming and are not able to handle the huge amount of transaction records. On the other side, most work employed hierarchical agglomerative clustering algorithm which are not scalable to large-scale transaction data [4]–[7]. In the past decades, researchers have proposed many clustering algorithms, such as spectral clustering [8], subspace clustering [9], nonnegative low-rank matrix factorization [10], spectral clustering [11], ensemble clustering [12], etc. However, few methods have been used for clustering transaction data.

Recently, Chen *et al.* [13], [14] proposed a PurTreeClust clustering algorithm for customer segmentation from large-scale transaction data. In this algorithm, a product tree is used to organize the categories in the transaction data, in which the leaf nodes denote products and the internal nodes represent product categories. A customer's transaction records can be compressed into a purchase tree, in which each product (item) bought by a customer corresponds to a leaf node in the product tree. Therefore, the set of customer transaction records are compressed into a set of purchase trees, in which each purchase tree represents a customer's purchase behavior. Then customer segmentation can be performed by clustering of these purchase trees. A PurTree distance metric was defined to compute the difference between two purchase trees, and a PurTreeClust algorithm was proposed to cluster purchase trees. The PurTreeClust algorithm first builds a cover tree for indexing the purchase tree data set, then selects initial cluster centers with a fast leveled density estimation method. Finally, the clustering result is obtained by assigning each customer to its nearest cluster center. However, it is difficult to adjust level weights in the PurTree distance and there is no optimization method for the clustering result. To conquer the above problem, Chen *et al.* [15] further proposed a local PurTree spectral clustering method, which can learn an adaptive similarity matrix from the local distances and the level weights in the PurTree distance simultaneously. In the new

Manuscript received November 24, 2017; revised February 22, 2018 and April 28, 2018; accepted May 9, 2018. This work was supported in part by NSFC under Grant 61773268, Grant 61732011, and Grant U1636202, and in part by Tencent Rhinoceros Birds-Scientific Research Foundation for Young Teachers of Shenzhen University. This paper was recommended by Associate Editor R. Tagliaferri. *(Corresponding author: Xiaojun Chen.)*

X. Chen, W. Sun, B. Wang, and X. Wang are with the College of Computer Science and Software, Shenzhen University, Shenzhen 518060, China (e-mail: xjchen@szu.edu.cn; 1329709650@qq.com; 412580829@qq.com; xizhaowang@ieee.org).

Z. Li is with the School of Computer Science and Engineering, University of New South Wales, Sydney, NSW 2052, Australia (e-mail: zhihuilics@gmail.com).

Y. Ye is with the Shenzhen Key Laboratory of Internet Information Collaboration, Shenzhen Graduate School, Harbin Institute of Technology, Shenzhen 518055, China (e-mail: yeyunming@hit.edu.cn).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCYB.2018.2836804

method, the node weights in the PurTree distance are set as equal for the children nodes of a parent node and a parameter γ is used to control the level weights β. However, since a product tree often consists of hundreds of thousands nodes, it is desired to learn a set of sparse node weights such that only a few important nodes are considered for computing the distance. Moreover, it is difficult to set proper γ.

In the past decade, soft subspace clustering has been an important research topic in cluster analysis. Many subspace clustering methods have been proposed for clustering of high-dimensional data, such as W-k-means [16], LAC [17], EWKM [18], ESSC [19], TW-k-means [20], FG-k-means [21], DSKmeans [22], and TWCC [23]. Such methods assign weights to individual variables and reduce the affection of noise features by assigning them with low weights, thus being useful for high-dimensional data. However, all above methods only work for flat features but cannot work for the tree structure features in the purchase tree data. Inspired by these work in subspace clustering, we propose to assign a set of sparse weights to the nodes in a purchase tree and learn these weights such that the important nodes are assigned with large weights while the noise nodes are assigned with small weights that are close to zero.

In this paper, we propose a two-level subspace weighting spectral clustering (TSW) algorithm to solve the above shortcomings. A PurTree subspace metric is proposed to measure the difference between two trees, in which a set of node weights are to be learned. The new model learns an adaptive similarity matrix and a set of sparse node weights simultaneously. We present an iterative optimization algorithm to optimize the proposed model, and an efficient method to compute a proper regularization parameter. Experimental results on ten benchmark data sets show the superior performance of TSW.

The rest of this paper is organized as follows. Notations and preliminaries are given in Section II. We present the PurTree subspace distance in Section III, and the TSW algorithm in Section V. The experimental results and analysis are presented in Section VI. The conclusions and future work are given in Section VII.

II. Notations and Preliminaries

In this section, we introduce the notations and preliminaries used in this paper.

A. Notations

Let T be a rooted tree with nodes $N(T)$ and edges $E(T) \subseteq N(T) \times N(T)$. The root of T is denoted by root(T). A node without children is a leaf node, and otherwise an internal node. For an edge $(u, v) \in E(T)$, node u is the parent node and v is the child node, i.e., $P_v(T) = u$ and $v \in C_u(T)$. The descendants of v, the nodes in all paths reachable from v to leaf nodes, is denoted as des(v). The level of a node is defined by 1 + (the number of edges between the node and the root node). $N^l(T)$ represents nodes in the lth level of T. The height of tree T, denoted by $H(T)$, is the number of edges on the longest downward path between the root node and a leaf node.

B. Product Tree and Purchase Tree

Let Ψ be a rooted tree used to systematically organize the items with multiple levels of categories, in which each leaf node represents an item and each internal node represents a category. All leaf nodes in Ψ are assumed to have equal depth in [13]. A purchase tree φ is used to illustrate the items bought by a customer, which is a subgraph of Ψ, i.e., $N(\varphi) \subseteq N(\Psi)$, $E(\varphi) \subseteq E(\Psi)$. Given a leaf node $u \in N(\varphi)$, the path from $root(\varphi)$ to u also exists in Ψ. For each purchase tree φ, we have $H(\varphi) = H(\Psi)$.

C. PurTree Metric Distance

Given a product tree Ψ and a set of n purchase trees $\Phi = \{\varphi_1, \ldots, \varphi_n\}$ where $\varphi_i \in \Psi$, let $H(\Phi)$ be the height of these purchase trees, root(φ) be the empty root node of the purchase tree. The PurTree distance is defined as follows [13].

Definition 1: Given a product tree Ψ and n purchase trees $\Phi = \{\varphi_1, \ldots, \varphi_n\}$ where $\varphi_i \in \Psi$, the PurTree distance between φ_i and φ_j is defined as

$$d(\varphi_i, \varphi_j) = \sum_{l=1}^{H(\Phi)} \beta_l \sum_{v \in N^l(\varphi_i) \cup N^l(\varphi_j)} \alpha_v \delta_v(\varphi_i, \varphi_j) \tag{1}$$

where $\delta_v(\varphi_i, \varphi_j)$ is the Jaccard distance of φ_i and φ_j on an internal node v, which is defined as

$$\delta_v(\varphi_i, \varphi_j) = 1 - \frac{\left|C_v(\varphi_i) \bigcap C_v(\varphi_j)\right|}{\left|C_v(\varphi_i) \bigcup C_v(\varphi_j)\right|} \tag{2}$$

and α_v is the node weight for node $v \in N(\Psi)$ which is defined as

$$\alpha_v = \begin{cases} 1 & \text{if } v = \text{root}(\Psi) \\ \frac{\alpha_u}{|C_u(\varphi_i) \bigcup C_u(\varphi_j)|} & \text{where } v \in C_u(\varphi_i) \bigcup C_u(\varphi_j) \end{cases} \tag{3}$$

and β_l is the lth level weight. $\{\beta_1, \ldots, \beta_{H(\Phi)}\}$ is a geometric sequence with common ratio γ ($\gamma >= 0$) under constraint $\sum_{l=1}^{H(\Phi)} \beta_l = 1$, defined as

$$\beta_l = \begin{cases} \frac{1-\gamma}{1-\gamma^{H(\Phi)}} \gamma^{l-1} & \text{for } \gamma > 0 \text{ and } \gamma \neq 1 \\ \frac{1}{H(\Phi)} & \text{for } \gamma = 1 \\ 1 & \text{for } \gamma = 0 \text{ and } l = 1 \\ 0 & \text{for } \gamma = 0 \text{ and } 1 < l \leq H(\Phi). \end{cases} \tag{4}$$

III. PurTree Subspace Metric

In (1), the node weights α are set as equal values for the children nodes of a parent node and a parameter γ is used to control the level weights β. However, since a product tree often consists of hundreds of thousands nodes, it is desired to learn a set of sparse node weights such that only a few important nodes are considered for computing the distance. Moreover, it is difficult to set proper γ. To solve above problems, we propose a new PurTree subspace distance which is defined as follows.

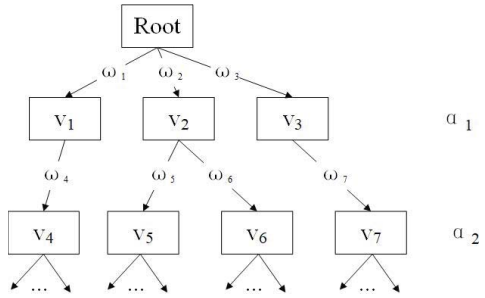

Fig. 1. Example of node weights and level weights in the PurTree subspace distance.

Definition 2: Given a product tree Ψ and a set of n purchase trees $\Phi = \{\varphi_1, \ldots, \varphi_n\}$, $\varphi_i \in \Psi$, the PurTree subspace distance between two purchase trees φ_i and φ_j is defined as

$$d(\varphi_i, \varphi_j, \alpha, \Omega) = \sum_{l=1}^{H(\Phi)} \alpha_l \sum_{v \in N^l(\varphi_i) \cup N^l(\varphi_j)} \omega_v \delta_v(\varphi_i, \varphi_j) \quad (5)$$

where $\delta_v(\varphi_i, \varphi_j)$ is defined in (2). $\alpha = \{\alpha_l\}_{l=1}^{H(\Psi)}$ is the level weights that sum to 1, in which α_l is the weight for the lth level. $\Omega = \{\omega_v | v \in N(\Phi)\}$ consists of $N(\Phi)$ node weights, in which ω_v is the weight for node v. Ω is defined under the following constraint:

$$\begin{cases} \forall v \in N(\Psi), \omega_v \in [0, 1] \\ \sum_{l=1}^{H(\Phi)} \sum_{v \in N^l(\Phi)} \omega_v = H(\Psi) \\ \sum_{t \in C_v(\Psi)} \omega_t = \omega_v. \end{cases} \quad (6)$$

Fig. 1 shows an example of node weights and level weights in the PurTree distance. The following theorem states the properties of Ω.

Theorem 1: $\forall 1 \le l \le H(\Phi)$, $\sum_{v \in N^l(\Phi)} \omega_v = 1$.

Proof: $\forall 1 \le l \le H(\Phi)$, we can verify that

$$\sum_{v \in N^l(\Phi)} \omega_v = \sum_{v \in N^l(\Phi)} \sum_{t \in C_v(\Psi)} \omega_t = \sum_{t \in N^{l+1}(\Psi)} \omega_t.$$

Then we have $\sum_{v \in N^l(\Phi)} \omega_v = 1$. ■

Theorem 2: Given two purchase trees φ_i, φ_j in Φ, $d(\varphi_i, \varphi_j, \Omega) \in [0, 1]$.

Proof: Since $\delta_v(\varphi_i, \varphi_j)$ lies in $[0, 1]$, we can verify that $d(\varphi_i, \varphi_j, \alpha, \Omega) \ge 0$ and

$$\begin{aligned} d(\varphi_i, \varphi_j, \alpha, \Omega) &= \sum_{l=1}^{H(\Phi)} \alpha_l \sum_{v \in N^l(\Phi)} \omega_v \delta_v(\varphi_i, \varphi_j) \\ &\le \sum_{l=1}^{H(\Phi)} \alpha_l \sum_{v \in N^l(\Phi)} \omega_v = \sum_{l=1}^{H(\Phi)} \alpha_l = 1 \end{aligned} \quad (7)$$

which completes the proof. ■

Theorem 3: $d(\varphi_i, \varphi_j, \alpha, \Omega)$ defined in (5) is a metric.

Proof: Since the Jaccard distance is a metric, we can verify that $d(\varphi_i, \varphi_j, \alpha, \Omega) = d(\varphi_j, \varphi_i, \alpha, \Omega)$ (symmetry), $d(\varphi_i, \varphi_i, \alpha, \Omega) = 0$ (reflexivity) and $d(\varphi_i, \varphi_j, \alpha, \Omega) \ge 0$ for all φ_i and φ_j in Ψ (positivity).

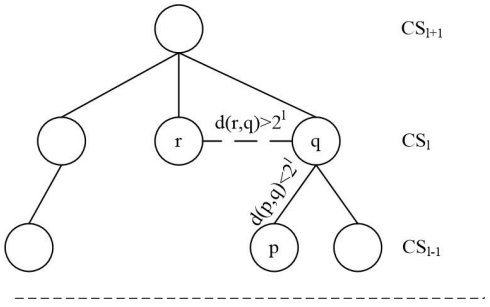

Fig. 2. Example of cover tree.

Given three purchase trees φ_i, φ_j, and φ_t

$$\begin{aligned} &d(\varphi_i, \varphi_j, \alpha, \Omega) + d(\varphi_j, \varphi_t, \Omega) \\ &\quad = \sum_{l=1}^{H(\Phi)} \alpha_l \sum_{v \in N^l(\varphi_i) \cup N^l(\varphi_j)} \omega_v \delta_v(\varphi_i, \varphi_j, \alpha, \omega) \\ &\qquad + \sum_{l=1}^{H(\Phi)} \alpha_l \sum_{v \in N^l(\varphi_j) \cup N^l(\varphi_t)} \omega_v \delta_v(\varphi_j, \varphi_t, \alpha, \omega) \\ &\quad = \sum_{l=1}^{H(\Phi)} \alpha_l \sum_{v \in N^l(\varphi_i) \cup N^l(\varphi_j)} \omega_v \left[\delta_v(\varphi_i, \varphi_j) + \delta_v(\varphi_j, \varphi_t)\right] \\ &\quad \ge \sum_{l=1}^{H(\Phi)} \alpha_l \sum_{v \in N^l(\varphi_i) \cup N^l(\varphi_t)} \omega_v \delta_v(\varphi_i, \varphi_t) \\ &\quad = d(\varphi_i, \varphi_t, \alpha, \Omega) \end{aligned} \quad (8)$$

which indicates that the new distance satisfies triangle inequality. Therefore, it is a metric. ■

IV. Fast k-Nearest Neighbor Search With Cover Tree

Cover tree is a leveled tree data structure for fast nearest neighbor operations in metric space [24]. Let CT be a cover tree on a data set $\mathbf{D}$, as shown in Fig. 2. Each level in CT is indexed by an integer scale l which increases from bottom to top. Denote the set of objects in $\mathbf{D}$ associated with the nodes at level l as CS_l. Here, each object in $\mathbf{D}$ appears at most once in each level, but may be associated with multiple nodes in the tree. CT obeys the following invariants for all levels.

1) *Nesting:* $CS_l \in CS_{l-1}$. If an object $p \in \mathbf{D}$ appears in CS_l, then p is associated with a node in each lower level.
2) *Covering:* For each object $p \in CS_{l-1}$, there exists a $q \in CS_l$ such that $d(p, q) < 2^l$ and the node in level l associated with q is a parent of the node in level $l-1$ associated with p.
3) *Separation:* For two different objects $r, q \in CS_l$, $d(r, q) > 2^l$, where d is a metric defined on $\mathbf{D}$. Let c be the *expansion constant* of S, which is a measure of the intrinsic dimensionality (as defined in [25]).

CT has the following properties.

1) The number of children of any node p is bounded by c^4.
2) The maximum depth of any point p is $O(c^2 \log(n))$.

To find the k-nearest neighbors of a point p from a cover tree CT, we descend through the tree from the top level to bottom level, keeping track of a subset $Q_i \in CS_i$ of nodes that may

Algorithm 1 Find-kNN (Cover Tree CT, Object p, Number of Nearest Neighbors k, and Base Parameter α)

1: **Input:** the nearest neighbors of p in CT.
2: Set $Q_\infty = C_\infty$, where C_∞ is the root level of CT.
3: Set $R = C_\infty$
4: **for** $i = \infty$ to -∞ **do**
5: Set $Q' = \{C_q : q \in Q_i\}$.
6: Set $Q = Q' \cup R$.
7: **if** $|Q| \geq k$ **then**
8: Form R with k nearest objects in Q, i.e., $|R| = \max k, |Q|$ and $\forall q \in R$ and $q' \in Q - R$, $d(p, q) \leq d(p, q')$.
9: $\beta = \max_{q \in R} d(p, q)$.
10: Form cover set $Q_{i-1} = \{q \in Q' : d(p, q) \leq \beta + 2^i\}$.
11: **else**
12: Form cover set $Q_{i-1} = \{q \in Q'\}$.
13: **end if**
14: **end for**
15: If $|Q| > k$, update Q by only retaining the k-nearest neighbor of p. Return Q as the ultimate result.

contain the k-nearest neighbors of p as descendants. The algorithm for finding the k-nearest neighbors of an object p from a cover tree CT is shown in Algorithm 1, which iteratively constructs Q_{i-1} by expanding Q_i to its children in CS_{i-1} and throwing away any child q that cannot lead to the k-nearest neighborhoods of p. Although the algorithm is stated using an infinite levels, it only needs to operate at most the height levels of CT.

Fig. 3 shows an example of 4-nearest neighbors search, in which nodes in light color represent the candidate k-nearest neighbors. Here, Q_i, Q, R, and β are variables in Algorithm 1.

The following theorem ensures the correctness of Algorithm 1.

Theorem 4: Given a cover tree CT built on $\mathbf{D}$, Find $-kNN(CT, p, k, \alpha)$ returns the exact k-nearest neighbors of p in $\mathbf{D}$.

Proof: For any $q \in CS_{i-1}$, the distance between q and any descendant q' is bounded by $d(q, q') \leq \sum_{j=i-1}^{-\infty} 2^j = 2^i$. If q is a k-nearest neighbor of p, we have $d(p, q) \leq \beta$. Since d is a metric, according to the triangularity, $d(p, q') \leq d(p, q) + d(q, q') \leq 2^i + \beta$. So Q_{i-1} in step 10 can never throw out a grandparent of the k-nearest neighbor of q. Eventually, k-nearest neighbors are in Q_i and step 15 returns the exact k-nearest neighbors of p. ■

V. Two-Level Subspace Weighting Spectral Clustering

To segment customers according to the customer transaction records, we first represent each customer as a purchase tree. Then customer segmentation can be performed by clustering of these purchase trees. Assume that we have obtained a product tree Ψ and n purchase trees $\Phi = \{\varphi_1, \ldots, \varphi_n\}$ in which φ_i is associated with the ith customer, we want to cluster Φ into c clusters, from which the customer segmentation result can be

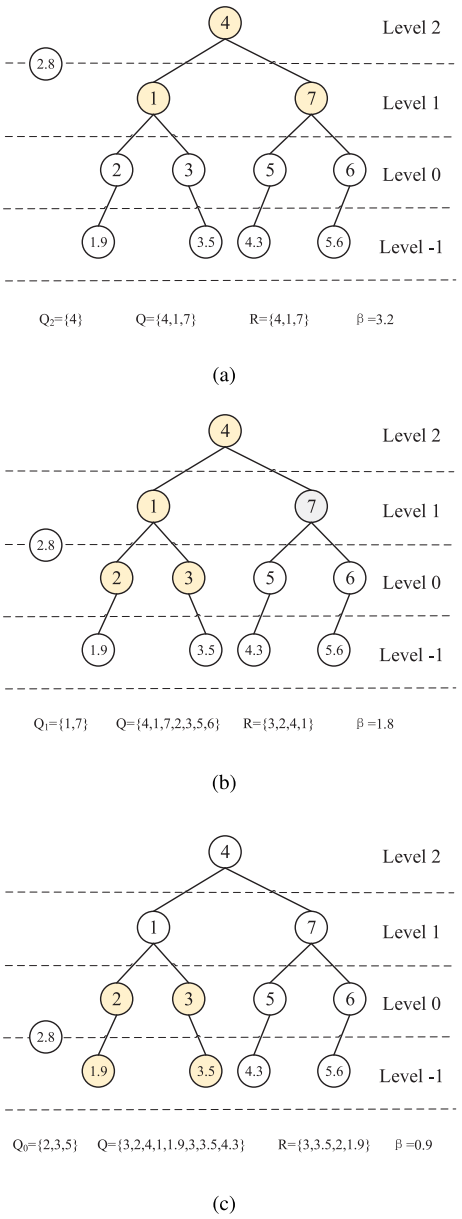

Fig. 3. Process of finding 4-nearest neighbors in the cover tree from top to bottom by Algorithm 1. Variables in searching the (a) level 1, (b) level 0, and (c) level -1.

obtained. Inspired by the work in [26], we learn a new graph represented by affinity matrix $\mathbf{P} = [p_{ij}]_{n \times n}$, in which p_{ij} is the probability that two trees φ_i and φ_j are connected. In order to find c clusters from Φ, we hope that the graph constructed from $\mathbf{P}$ only consists of c connected components. To achieve this goal, we present the following problem to simultaneously learn the node weights ω, level weights α, and the connection probability matrix $\mathbf{P}$

$$\min_{\mathbf{P}, \Omega} \sum_{i,j=1}^{n} \left[\sum_{l=1}^{H(\Phi)} \alpha_l \sum_{v \in N^l(\Phi)} \omega_v d_{ij}^v p_{ij} + \lambda p_{ij}^2 \right] + \theta \sum_{l=1}^{H(\Psi)} \alpha_l^2 + \eta \sum_{l=1}^{H(\Phi)} \sum_{v \in N^l(\Phi)} \omega_v^2 \quad (9)$$

$$\text{subject to } \begin{cases} \forall i, \mathbf{p}_i^T \mathbf{1} = 1, & p_{ij} \in [0,1] \\ \alpha^T \mathbf{1} = 1, & \alpha_l \in [0,1] \\ \sum_{l=1}^{H(\Phi)} \sum_{v \in N^l(\Phi)} \omega_v = H(\Psi), & \omega_v \in [0,1] \\ \sum_{t \in C_v(\Psi)} \omega_t = \omega_v & \\ \text{rank}(\mathbf{L}_P) = n - c & \end{cases} \tag{10}$$

where d_{ij}^v is the abbreviation of $\delta_v(\varphi_i, \varphi_j)$ which is computed according to (2), $\mathbf{L}_P = \mathbf{D}_P - [(\mathbf{P}^T + \mathbf{P})/2]$ is the Laplacian matrix, the degree matrix $\mathbf{D}_P \in \mathbb{R}^{n \times n}$ is defined as a diagonal matrix in which $d_{ij} = \sum_{j=1}^{n} [(p_{ij} + p_{ji})/2]$, λ and η are two regularization parameters.

The first term in (9) is the pairwise product of the PurTree subspace distance and connection probability, in which a smaller distance $d(\varphi_i, \varphi_j, \Omega)$ is assigned with a larger probability p_{ij}. The second and third terms are regularization terms of $\mathbf{P}$ and Ω, where λ and η are two regularization parameters. The rank constraint $\text{rank}(\mathbf{L}_P) = n - c$ is imposed to $\mathbf{L}_P$, such that the sparse graph constructed from P only consists of c connected components [27], [28]. The final clustering results can be obtained by assigning the objects in the same connected component in the graph constructed from $\mathbf{P}$ to the same cluster.

On the other hand, the constraint $\text{rank}(\mathbf{L}_P) = n - c$ is equivalent to the following problem [26]:

$$\min_{\mathbf{F} \in \mathbb{R}^{n \times c}, \mathbf{F}^T \mathbf{F} = I} 2\mu \text{Tr}\big(\mathbf{F}^T \mathbf{L}_P \mathbf{F}\big) \tag{11}$$

where μ is a large enough parameter.

Then we reform problem (9) as the following problem:

$$\begin{aligned} \min \quad & \sum_{i,j=1}^{n} \left[\sum_{l=1}^{H(\Phi)} \alpha_l \sum_{v \in N^l(\Phi)} \omega_v d_{ij}^v p_{ij} + \lambda p_{ij}^2 \right] + \theta \sum_{l=1}^{H(\Psi)} \alpha_l^2 \\ & + \eta \sum_{l=1}^{H(\Phi)} \sum_{v \in N^l(\Phi)} \omega_v^2 + 2\mu T_r\big(\mathbf{F}^T \mathbf{L}_P \mathbf{F}\big) \end{aligned} \tag{12}$$

$$\text{subject to } \begin{cases} \forall i, \mathbf{p}_i^T \mathbf{1} = 1, & p_{ij} \in [0,1] \\ \alpha^T \mathbf{1} = 1, & \alpha_l \in [0,1] \\ \sum_{l=1}^{H(\Phi)} \sum_{v \in N^l(\Phi)} \omega_v = H(\Psi), & \omega_v \in [0,1] \\ \sum_{t \in C_v(\Psi)} \omega_t = \omega_v & \\ \mathbf{F}^T \mathbf{F} = I, & \mathbf{F} \in \mathbb{R}^{n \times c}. \end{cases} \tag{13}$$

We can apply the alternative optimization approach to solve problem (12). In the following, we show how to update each of four variables.

A. Update $\mathbf{F}$ With $\mathbf{P}$, α, and Ω Fixed

When $\mathbf{P}$, α, and Ω are fixed, problem (12) becomes

$$\min_{\mathbf{F} \in \mathbb{R}^{n \times c}, \mathbf{F}^T \mathbf{F} = I} \text{Tr}\big(\mathbf{F}^T \mathbf{L}_P \mathbf{F}\big). \tag{14}$$

The optimal solution $\mathbf{F}$ in problem (14) is formed by c eigenvectors of $\mathbf{L}_P$ which corresponds to its c smallest eigenvalues [26].

B. Update $\mathbf{P}$ With $\mathbf{F}$, α, and Ω Fixed

When $\mathbf{F}$, α, and Ω are fixed, problem (12) becomes

$$\begin{aligned} \min \quad & \sum_{i,j=1}^{n} \sum_{l=1}^{H(\Phi)} \sum_{v \in N^l(\Phi)} \omega_v d_{ij}^v p_{ij} + \lambda \sum_{i}^{n} \sum_{j \in \mathcal{M}_i} p_{ij}^2 \\ & + 2\mu \text{Tr}\big(\mathbf{F}^T \mathbf{L}_P \mathbf{F}\big) \\ \text{s.t.} \quad & \mathbf{P}, \forall i, \sum_{j \in \mathcal{M}_i} p_{ij} = 1, p_{ij} \in [0,1]. \end{aligned} \tag{15}$$

It can be verified that

$$2\text{Tr}\big(\mathbf{F}^T \mathbf{L}_P \mathbf{F}\big) = \sum_{i=1}^{n} \sum_{j \in \mathcal{M}_i} p_{ij} \|f_i - f_j\|_2^2$$

where $f_i \in \mathbb{R}^{c \times 1}$ is the transpose of the ith row of F. Then problem (12) can be rewritten as

$$\begin{aligned} \min_{\mathbf{P}} \quad & \sum_{i,j=1}^{n} \left[\left(\sum_{l=1}^{H(\Phi)} \alpha_l \sum_{v \in N^l(\Phi)} \omega_v d_{ij}^v + \mu \|f_i - f_j\|_2^2 \right) p_{ij} + \lambda p_{ij}^2 \right] \\ \text{s.t.} \quad & \forall i, \sum_{j \in \mathcal{M}_i} p_{ij} = 1, p_{ij} \in [0,1]. \end{aligned} \tag{16}$$

Since problem (16) is independent between different $\mathbf{p}_i$, we can solve it individually for each $\mathbf{p}_i$ (the vector form of $[p_{i1}, \ldots, p_{in}]$). Denote $d_{ij}^f = \|f_i - f_j\|_2^2$, $d_{ij}^\Psi = \sum_{l=1}^{H(\Phi)} \alpha_l \sum_{v \in N^l(\Phi)} \omega_v d_{ij}^v$, and $d_i \in \mathbb{R}^{n \times 1}$ as a vector with the jth element as $d_{ij} = d_{ij}^\Psi + \mu d_{ij}^f$.

The Lagrangian function of problem (16) for $\mathbf{p}_i$ is

$$\begin{aligned} \mathcal{L}(\mathbf{p}_i, \chi, \tau) = & \lambda \sum_{j \in \mathcal{M}_i} p_{ij}^2 + \sum_{j \in \mathcal{M}_i} d_{ij} p_{ij} \\ & + \chi \left(\sum_{j \in \mathcal{M}_i} p_{ij} - 1 \right) - p_i^T \tau \end{aligned} \tag{17}$$

where χ and positive vector τ are Lagrangian multipliers.

It can be verified that the optima p_{ij}^* is

$$p_{ij}^* = \left(-\frac{1}{2\lambda} \big[d_{ij} + \chi^* \big] \right)_+ \tag{18}$$

where $a_+ = \max(a, 0)$. So we can obtain the optimal solution of p_{ij}^* if we know χ^*. Since $\sum_{j \in \mathcal{M}_i} p_{ij}^* = 1$, we can obtain χ^* by solving the following root finding problem:

$$f(\chi^*) = \sum_{j \in \mathcal{M}_i} \left(-\frac{1}{2\lambda} \big[d_{ij} + \chi^* \big] \right)_+ - 1 = 0. \tag{19}$$

Note that $\chi^* \geq 0$, $f'(\chi^*) \leq 0$ and $f'(\chi^*)$ is a piecewise linear and convex function, we can use the Newton method to find the root of $f(\chi^*) = 0$.

C. Update α With $\mathbf{P}$, $\mathbf{F}$, and Ω Fixed

When $\mathbf{P}$, $\mathbf{F}$, and Ω are fixed, problem (12) becomes

$$\begin{aligned} \min \quad & \sum_{l=1}^{H(\Psi)} \sum_{i,j=1}^{n} \sum_{v \in N^l(\Psi)} \omega_v d_{ij}^v p_{ij} \alpha_l + \theta \sum_{l=1}^{H(\Psi)} \alpha_l^2 \\ \text{s.t.} \quad & \alpha^T \mathbf{1} = 1, \alpha_l \in [0,1]. \end{aligned} \tag{20}$$

Let $\mathbf{E}_l = \sum_{i,j=1}^{n} \sum_{v \in N^l(\Psi)} \omega_v d_{ij}^v p_{ij}$. The Lagrangian function of problem (20) for α is

$$\mathcal{L}(\alpha, \chi, \tau) = \sum_{l=1}^{H(\Psi)} \mathbf{E}_l \alpha_l + \theta \sum_{l=1}^{H(\Psi)} \alpha_l^2 + \kappa \left(\sum_{l=1}^{H(\Psi)} \alpha_l - 1 \right) - \alpha^T \tau \tag{21}$$

where κ and positive vector τ are Lagrangian multipliers.

According to the KKT condition [29], we have

$$\begin{cases} \frac{\partial \mathcal{L}(\omega_v | v \in N^l(\Psi))}{\partial \alpha_l} = E_l + 2\theta^T \alpha + \kappa = 0 \\ \frac{\partial \mathcal{L}(\omega_v | v \in N^l(\Psi))}{\partial \chi} = \sum_{l=1}^{H(\Psi)} \alpha_l - 1 = 0 \\ \forall v \in C^t, \tau_v = 0. \end{cases} \tag{22}$$

It can be verified that the optima α_l can be obtained by solving the following problem:

$$\begin{cases} \alpha_l = \left(-\frac{1}{2\theta} \left[\sum_{i,j=1}^{n} \sum_{v \in N^l(\Psi)} \omega_v d_{ij}^v p_{ij} + \kappa \right] \right)_+ \\ \sum_{l=1}^{H(\Psi)} \alpha_l = 1. \end{cases} \tag{23}$$

It can be verified that the optima α_l^* is

$$\alpha_l^* = \left(-\frac{1}{2\theta} \left[\sum_{i,j=1}^{n} \sum_{v \in N^l(\Psi)} \omega_v d_{ij}^v p_{ij} + \kappa^* \right] \right)_+ \tag{24}$$

where $a_+ = \max(a, 0)$. So we can obtain the optimal solution of α_l^* if we know κ^*. Since $\sum_{l=1}^{H(\Psi)} \alpha_l^* = 1$, we can obtain κ^* by solving the following root finding problem:

$$f(\kappa^*) = \sum_{l=1}^{H(\Psi)} \left(-\frac{1}{2\theta} \left[\sum_{i,j=1}^{n} \sum_{v \in N^l(\Psi)} \omega_v d_{ij}^v p_{ij} + \kappa^* \right] \right)_+ - 1 = 0. \tag{25}$$

Note that $\kappa^* \geq 0$, $f'(\kappa^*) \leq 0$ and $f'(\kappa^*)$ is a piecewise linear and convex function, we can use the Newton method to find the root of $f(\kappa^*) = 0$.

D. Update Ω With $\mathbf{P}$, $\mathbf{F}$, and α Fixed

When $\mathbf{P}$, $\mathbf{F}$, and α are fixed, problem (12) becomes

$$\begin{aligned} \min \quad & \sum_{i,j=1}^{n} \sum_{l=1}^{H(\Phi)} \alpha_l \sum_{v \in N^l(\Phi)} \omega_v d_{ij}^v p_{ij} \\ & + \eta \sum_{l=1}^{H(\Phi)} \alpha_l \sum_{t \in N^l(\Phi)} \sum_{v \in C_t} \omega_v^2 \end{aligned} \tag{26}$$

$$\text{subject to} \quad \begin{cases} \sum_{l=1}^{H(\Phi)} \sum_{v \in N^l(\Phi)} \omega_v = H(\Psi), \quad \omega_v \in [0, 1] \\ \sum_{t \in C_v(\Psi)} \omega_t = \omega_v. \end{cases} \tag{27}$$

According to Theorem 1, the sum of node weights in each level is a constant value 1. Therefore, we can sequently optimize the node weights level by level. We first let $\omega_{\text{Root}} = 1$. Given an internal node $t \in N^l(\Phi)$, we fix ω_t and solve the following problem for $\{\omega_v | v \in C_t\}$:

$$\min_{\sum_{v \in C_t} \omega_v = \omega_t} \sum_{i,j=1}^{n} \sum_{v \in C_t} \omega_v d_{ij}^v p_{ij} + \eta \sum_{v \in C_t} \omega_v^2. \tag{28}$$

The Lagrangian function of problem (28) is

$$\begin{aligned} \mathcal{L}(\omega_v | v \in C_t) = & \sum_{v \in C_t} \omega_v \sum_{i,j=1}^{n} d_{ij}^v p_{ij} + \eta \sum_{v \in C_t} \omega_v^2 \\ & + \zeta_t \left(\sum_{v \in C_t} \omega_v - \omega_t \right) - \sum_{v \in C_t} \tau_v \omega_v \end{aligned} \tag{29}$$

where $t = P^v$, $\zeta_t \geq 0$ and $\{\tau_v \geq 0 | v \in C_t\}$ are multipliers.

It can be verified that the optima ω_v^* is

$$\omega_v^* = \left(-\frac{1}{2\eta} \left[\sum_{i,j=1}^{n} d_{ij}^v p_{ij} + \zeta_t^* \right] \right)_+ \tag{30}$$

where t is the children of v and $a_+ = \max(a, 0)$. So we can obtain the optimal solution of ω_v^* if we know ζ_t^*. Since $\sum_{v \in C_t} \omega_v = \omega_t$, we can obtain ω_v^* by solving the following root finding problem:

$$f(\zeta^*) = \sum_{v \in C_t} \left(-\frac{1}{2\eta} \left[\sum_{i,j=1}^{n} d_{ij}^v p_{ij} + \zeta_t^* \right] \right)_+ - \omega_t = 0. \tag{31}$$

Note that $\zeta^* \geq 0$, $f'(\zeta^*) \leq 0$ and $f'(\zeta^*)$ is a piecewise linear and convex function, we can use the Newton method to find the root of $f(\zeta^*) = 0$.

E. Determination of λ

Since a transaction data set often consists of a large number of users, it is useful to explore the local connectivity during the clustering process. Assuming that $\mathbf{P}$ is a sparse graph in which each p_i contains at most k positive values for its k-nearest neighbors. However, it is impossible to select the real k-nearest neighbors at the start because Ω is unknown. To simplify the computation, we select approximate k-nearest neighbors at the start, where the node weights $\omega_v \in \Omega$ are defined as

$$\omega_v = \begin{cases} \frac{1}{H(\Phi)} & \text{if } v = \text{Root}(\Psi) \\ \frac{\omega_t}{|C_t|} & \text{if } v \in C_t. \end{cases} \tag{32}$$

Then we from a partially ordered set $\{d_{ij}^e | 1 \leq j \leq n, j \neq i\}$, in which d_{ij}^e is the jth small value. The k-nearest neighbors $\mathcal{M}_i$ for φ_i can be formed by selecting k smallest values in $\{d_{ij}^e | 1 \leq j \leq n, j \neq i\}$. According to (24), we know that p_{ij} is negatively proportional to d_{ij}. If we wish p_i contains at most k positive values, we can set a proper λ such that only k-nearest neighbors $\mathcal{M}_i$ for φ_i are with positive p_{ij}. According to (24), we know that

$$\begin{cases} -\frac{d_{ik} + \chi}{2\lambda_i} > 0 \\ -\frac{d_{i,k+1} + \chi}{2\lambda_i} \leq 0 \end{cases} \tag{33}$$

where d_{ik} is the kth smallest distance to φ_i.

According to the constraint $\sum_{j \in \mathcal{M}_i} p_{ij} = 1$, we have

$$\chi = -\frac{2\lambda_i}{k} - \frac{1}{k} \sum_{j \in \mathcal{M}_i} d_{ij}. \tag{34}$$

So we have the following inequality for λ_i according to (33) and (34):

$$\frac{k}{2}d_{ik} - \frac{1}{2}\sum_{j\in\mathcal{M}_i} d_{ij} < \lambda_i \leq \frac{k}{2}d_{i,k+1} - \frac{1}{2}\sum_{j\in\mathcal{M}_i} d_{ij}. \quad (35)$$

In problem (9), we wish p_i contains exactly k positive values. Therefore, λ can be set as its upper bound. However, it is impossible to compute the exact upper bound λ_i since the node weights ω change in each iteration. We turn to compute an approximate upper bound of λ with equal level weights of ω, and set λ_i as the approximate upper bound

$$\lambda_i = \frac{k}{2}d^e_{i,k+1} - \frac{1}{2}\sum_{j\in\mathcal{M}_i} d^e_{ij} \quad (36)$$

where d^e_{ij} is the PurTree subspace distance computed with Ω defined in (32).

The overall λ can be set to the mean of $\{\lambda_1, \ldots, \lambda_n\}$

$$\lambda = \frac{1}{2n}\sum_{i=1}^{n}\left(kd^e_{i,k+1} - \sum_{j\in\mathcal{M}_i} d^e_{ij}\right). \quad (37)$$

According to (24), the optimal solution of p_{ij} for k-nearest neighbors is

$$p_{ij} = \left(-\frac{1}{2\lambda}\left[d_{ij} + \chi\right]\right)_+ \quad (38)$$

where χ can be computed from $\sum_{j\in\mathcal{M}_i} p_{ij} = 1$ and all d^f_{ij} are set as 0.

F. Optimization Algorithm

The detailed algorithm to solve the problem (12), named TSW, is summarized in Algorithm 2.[1] Given a set of n purchase trees $\Phi = \{\varphi_1, \ldots, \varphi_n\}$, we want to cluster Φ into c clusters. $\mathbf{F}$, Ω and $\mathbf{P}$ in (12) are iteratively solved until problem (12) converges. The final clustering result can be obtained from the connected components in the graph constructed from $\mathbf{P}$. If the algorithm needs r iterations to converge, the complexity of the TSW algorithm is $O(rkn^2)$.

VI. Experimental Results and Analysis

In this section, we report the experimental results on ten real-life transaction data sets and investigate the effectiveness and scalability of TSW.

A. Benchmark Data and Evaluation Methods

Ten real-life transaction data sets were used to investigate the effectiveness and scalability of the proposed algorithm. D_1 was built from a superstore's transactional data set,[2] which consists of 8399 transaction records from 796 customers. D_2–D_6 were built from five subsets of a super market's transactional data, which contains up to 25 million newest transaction records. D_7–D_{10} were built from four subsets of a year of purchase history transaction data from the kaggle competition,[3] which contains more than 349 million transaction records. All data sets were converted to purchase tree data sets with the code in https://github.com/xjchensz/PTREE. The characteristics of ten data sets are shown in Table I, in which D_2–D_6 contain five levels and other five data sets contain four levels.

TABLE I
Characteristics of Ten Customer Transaction Data Sets

Data sets (D)	Size	$\lvert N^2(D)\rvert$	$\lvert N^3(D)\rvert$	$\lvert N^4(D)\rvert$	$\lvert N^5(D)\rvert$
D_1	795	3	17	1264	
D_2	208	84	501	1198	9760
D_3	416	90	552	1339	14274
D_4	832	90	606	1457	18137
D_5	1665	91	651	1584	22822
D_6	3330	91	697	1714	28065
D_7	608	821	73164	89665	
D_8	1216	826	74500	91785	
D_9	2433	825	75737	93731	
D_{10}	4867	827	77424	96538	

Algorithm 2 TSW Algorithm to Solve Problem (12)

Input: A set of n purchase trees $\Phi = \{\varphi_1, \ldots, \varphi_n\}$, two regularization parameters θ and η, the number of nearest neighbors k, and the number of clusters c.
Initialize Ω according to Eq. (32) and find k nearest neighbors $\mathcal{M}_i$ for each φ_i with Algorithm 1, in which the node weights in the PurTree subspace distance defined in Eq. (32).
Initialize λ according to Eq. (37).
Initialize $\mathbf{P}$ according to Eq. (38), in which d^f_{ij} is set as 0.
repeat
  Update $\mathbf{F}$ by selecting c eigenvector of $\mathbf{L}_P = \mathbf{D}_P - \frac{\mathbf{P}^T+\mathbf{P}}{2}$ which corresponds to its c smallest eigenvalues.
  Update α according to Eq. (23).
  for $l = 2$ to $H(\Psi)$ **do**
    for $\forall\ v \in N^l(\Phi)$ **do**
      Update ω according to Eq. (30).
    end for
  end for
  Update $\mathbf{P}$ according to Eq. (38).
until (converges)
Output: Find the connected components in the graph constructed from $\mathbf{P}$, and assign the objects in the same connected component to the same cluster.

Since the ten data sets in Table I contain no labels, we use internal validation method to evaluate the clustering results. Given c clusters $\mathcal{C}$, the commonly used internal validation method $\log(W_k) = \log\{\sum_{l=1}^{c}(1/[2|\mathcal{C}_l|])\sum_{i,j\in\mathcal{C}_l} d(\varphi_i, \varphi_j)\}$. However, $\log(W_k)$ is affected by c and the weights in both PurTree distance and PurTree subspace distance. We may get different $\log(W_k)$ even for the same clustering result, with different node weights ω. So we normalize $\log(W_k)$ with the sum of all distances. In this paper, we use the normalized $\log(W_k)$

[1] In practice, μ can be set according to the method used in [26].
[2] https://community.tableau.com/docs/DOC-1236
[3] http://www.kaggle.com/c/acquire-valued-shoppers-challenge/data

TABLE II
PERFORMANCE COMPARISON OF $\mathcal{NLW}$ (*Mean* $\pm$ *Standard Deviation*) BY SEVEN CLUSTERING ALGORITHMS ON TEN BENCHMARK DATA SETS (THE BEST RESULT ON EACH DATA SET IS HIGHLIGHTED IN BOLD)

Data	DBSCAN	HAC	NCut	RCut	PurTreeClust	LPS	TSW
D_1	-7.47 ± 1.23	-7.75 ± 1.24	-7.41 ± 0.24	-7.41 ± 0.24	-7.83 ± 0.80	-7.60 ± 0.36	**-9.84** ± 0.80
D_2	-6.03 ± 0.04	-6.34 ± 0.90	-6.17 ± 0.40	-6.19 ± 0.40	-6.31 ± 0.80	-6.32 ± 0.35	**-6.59** ± 0.21
D_3	-6.73 ± 0.04	-6.94 ± 0.64	-6.81 ± 0.20	-6.81 ± 0.20	-6.91 ± 0.64	-7.02 ± 0.32	**-7.09** ± 0.21
D_4	-7.42 ± 0.04	-7.59 ± 0.63	-7.48 ± 0.18	-7.48 ± 0.24	-7.62 ± 0.52	-7.58 ± 0.36	**-7.72** ± 0.2
D_5	-8.11 ± 0.08	-8.26 ± 0.45	-8.16 ± 0.15	-8.15 ± 0.15	-8.34 ± 0.38	-8.38 ± 0.33	**-8.83** ± 0.25
D_6	-8.80 ± 0.08	-8.92 ± 0.58	-8.84 ± 0.12	-8.84 ± 0.12	-8.89 ± 0.41	-9.02 ± 0.34	**-9.67** ± 0.34
D_7	-7.10 ± 0.08	-7.19 ± 0.16	-7.17 ± 0.12	-7.16 ± 0.12	-7.40 ± 0.49	-7.62 ± 0.35	**-8.08** ± 0.46
D_8	-7.80 ± 0.08	-7.85 ± 0.14	-7.83 ± 0.12	-7.83 ± 0.12	-7.90 ± 0.47	-8.08 ± 0.34	**-8.73** ± 0.49
D_9	-8.49 ± 0.08	-8.53 ± 0.13	-8.52 ± 0.12	-8.51 ± 0.12	-8.58 ± 0.49	-8.58 ± 0.38	**-9.35** ± 0.48
D_{10}	-9.18 ± 0.08	-9.21 ± 0.12	-9.20 ± 0.08	-9.20 ± 0.08	-9.30 ± 0.50	-9.4 ± 0.35	**-9.90** ± 0.45

for evaluating a clustering result, which is computed as [15]

$$\mathcal{NLW}(\mathcal{C}) = \log\left\{\sum_{l=1}^{c}\frac{1}{2|\mathcal{C}_l|}\sum_{i,j\in\mathcal{C}_l} d(\varphi_i,\varphi_j)\right\} - \log\left(\sum_{i,j=1}^{n} d(\varphi_i,\varphi_j)\right) \tag{39}$$

where $\mathcal{C}$ consists of c clusters. The lower the $\mathcal{NLW}(\mathcal{C})$, the better the clustering result.

B. Results and Analysis

We used all ten data sets to compare the effectiveness of the TSW algorithm with six clustering algorithms, i.e., DBSCAN, HAC, Ncut [30], RCut [31], PurTreeClust [13], and LPS [15]. In this experiment, we selected five integers $\{10, 20, 30, 40, 50\}$ for c. The similarity matrices for DBSCAN, HAC, Ncut, RCut, and PurTreeClust were computed as 1 minus the PurTree distance matrices, in which γ was set as the same values used in [13], i.e., $\gamma = \{0, 0.2, 0.8, 1, 2, 1000\}$. Twenty integers from 5 to 100 were used for k in DBSCAN, PurTreeClust, LPS and TSW. The other parameters of all methods were set in the same strategy to make the experiments fair enough, i.e., $\{10^{-3}, 10^{-2}, 10^{-1}, 10^{0}, 10^{1}, 10^{2}, 10^{3}\}$. For DBSCAN, we also set eps as 95 values from 0.25 to 0.5 and minPts as 95 values from 1% to 10% of the number of objects. For each clustering algorithm, we computed the average $\mathcal{NLW}$ from the clustering results. The results are shown in Table II, in which the mean and standard deviation of $\mathcal{NLW}$ are reported. From these results, we can see that TSW produced the highest $\mathcal{NLW}$ on eight data sets. For example, on D_7–D_{10} which consists of more than 80 thousands of products, TSW achieves a greater than 7% average improvement compared with the second-best method LPS. On D_1, TSW even achieves a nearly 26% average improvement compared with the second-best method PurTreeClust. Besides TSW, LPS produced better results than the other methods.

We select a clustering result of TSW with 20 clusters on D_4 to check whether the uncovered clusters are meaningful. We first compute the average sparsity of the purchase trees in 20 clusters, where the sparsity of the lth level in φ_i with respect to the product tree Ψ is defined as $\mathcal{S}^l_\Psi(\varphi_i) = [(|N^l(\varphi_i)|)/(|N^l(\Psi)|)]$ [14]. Then the results are drawn in Fig. 4. From this figure, we can observe that the average sparsity of the 20 clusters are very different which indicates that the 20 clusters consist of different Purchase tree structures.

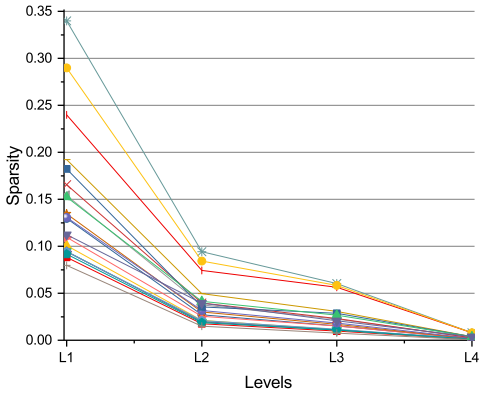

Fig. 4. Average sparsity of purchase trees in each of 20 clusters uncovered by TSW on D_4.

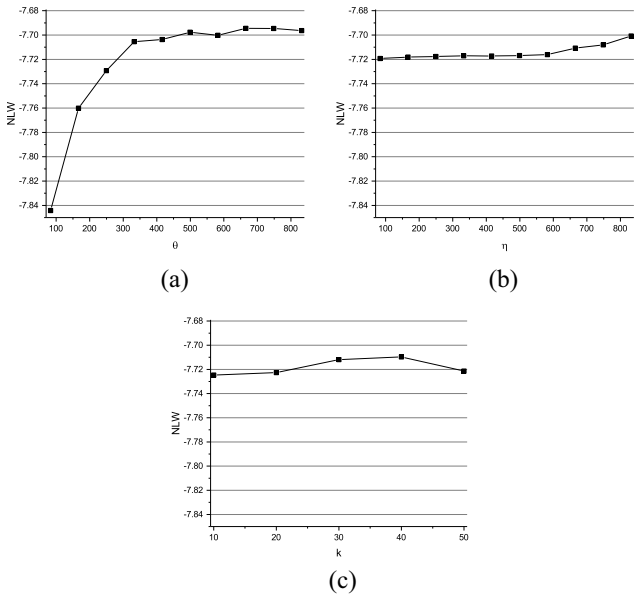

Fig. 5. *NLW* versus (a) θ, (b) η, and (c) k on D_4.

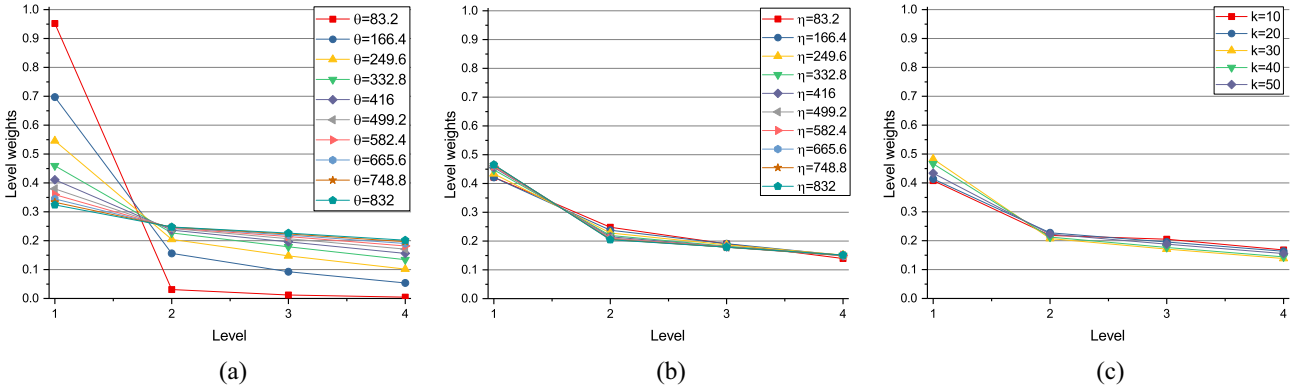

Fig. 6. Level weights versus (a) θ, (b) η, and (c) k on D_4.

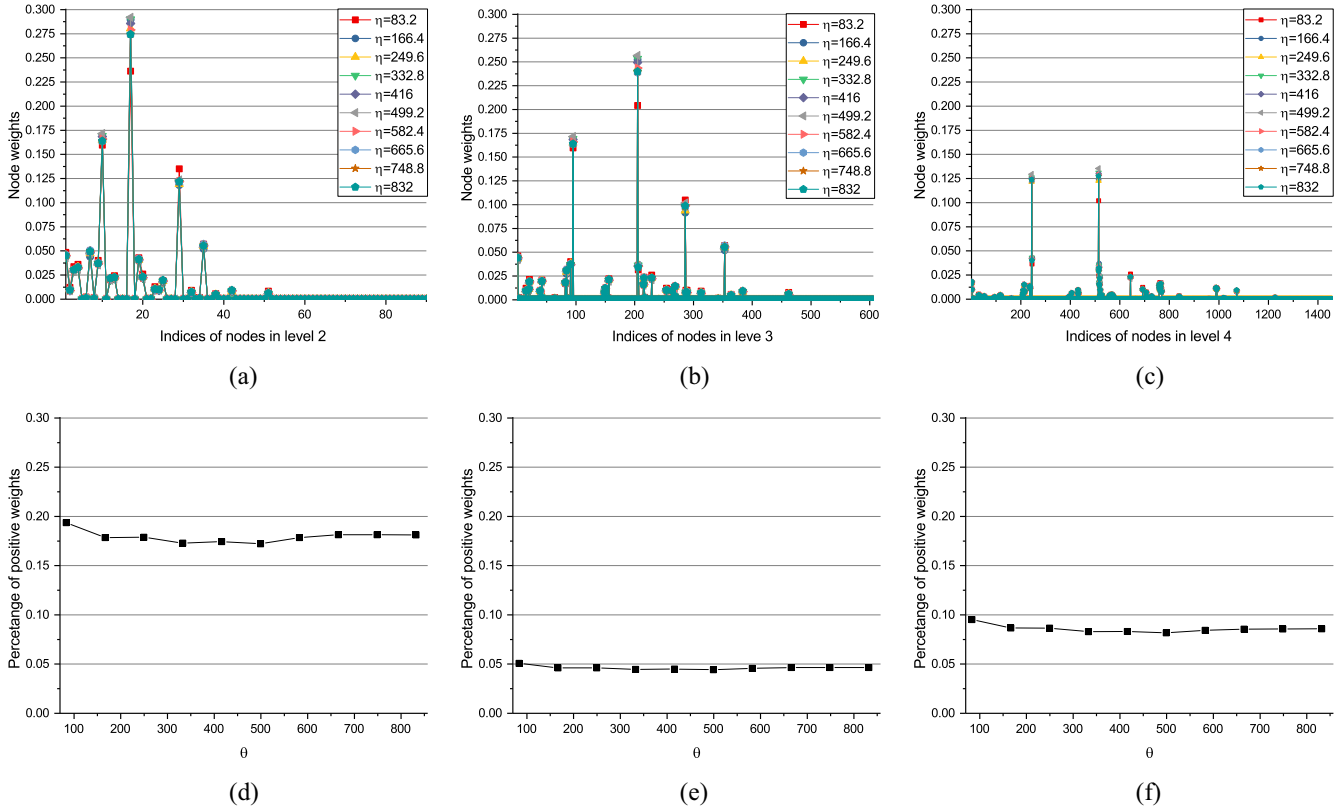

Fig. 7. Node weights Ω versus θ on D_4. (a) $\{\omega_v | v \in N^2(\Phi)\}$ versus θ. (b) $\{\omega_v | v \in N^3(\Phi)\}$ versus θ. (c) $\{\omega_v | v \in N^4(\Phi)\}$ versus θ. (d) Percentages of positive $\{\omega_v | v \in N^2(\Phi)\}$ versus θ. (e) Percentages of positive $\{\omega_v | v \in N^3(\Phi)\}$ versus θ. (f) Percentages of positive $\{\omega_v | v \in N^4(\Phi)\}$ versus θ.

To investigate the buying habits in difference clusters, we merge all purchase trees in a cluster into a purchase tree and study their differences. Experimental results show that some customer group prefer buying sport equipments, some customer group prefer buying domestic appliances, and some customer group prefer buying different kind of foods.

To study the important products in distinguishing the cluster structure, we select the important products from a clustering result according to the learned node weights by setting a proper threshold σ to form the important node set $\{v | \omega_v \geq \sigma\}$. Table III shows some samples of the most important products selected from the clustering results of TSW on D_4 according the average node weights.

C. Parameter Sensitivity Study

We select D_4 to analyze the change of $\mathcal{NLW}$ with three parameters θ, η, and k. The relationships between the average $\mathcal{NLW}$ and three parameters are shown in Fig. 5. From these figures, we can see that the average $\mathcal{NLW}$ are mainly affected by θ, and it is nearly stable with the change of k. $\mathcal{NLW}$ decrease with the decrease of θ. In such case, important levels will be assigned to bigger level weights. $\mathcal{NLW}$ also decreases as η decrease. According to (30), we know that more nodes will be assigned to positive node weights as η increases. This indicates that the learned level weights and sparse node weights indeed improve the clustering performance. We also observe that θ has greater influence in

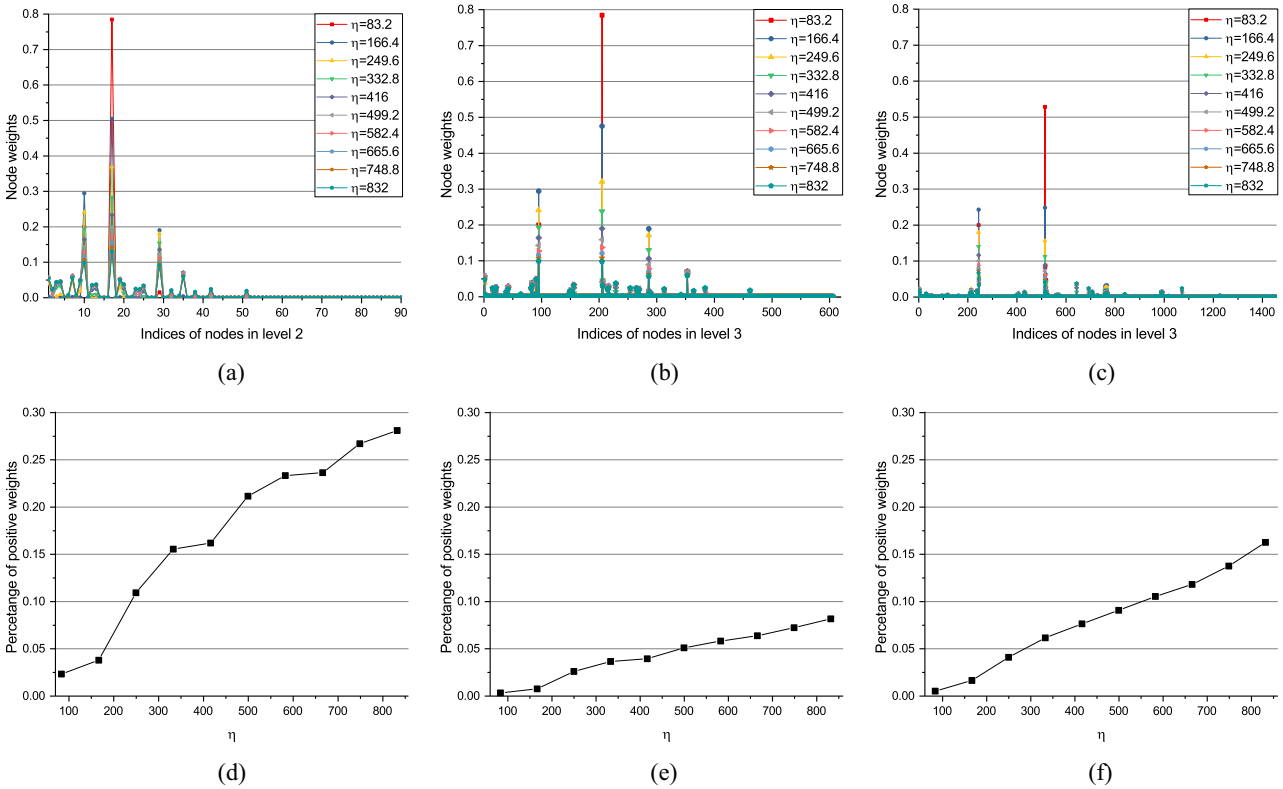

Fig. 8. Node weights Ω versus η on D_4. (a) $\{\omega_v | v \in N^2(\Phi)\}$ versus η. (b) $\{\omega_v | v \in N^3(\Phi)\}$ versus η. (c) $\{\omega_v | v \in N^4(\Phi)\}$ versus η. (d) Percentages of positive $\{\omega_v | v \in N^2(\Phi)\}$ versus η. (e) Percentages of positive $\{\omega_v | v \in N^3(\Phi)\}$ versus η. (f) Percentages of positive $\{\omega_v | v \in N^4(\Phi)\}$ versus η.

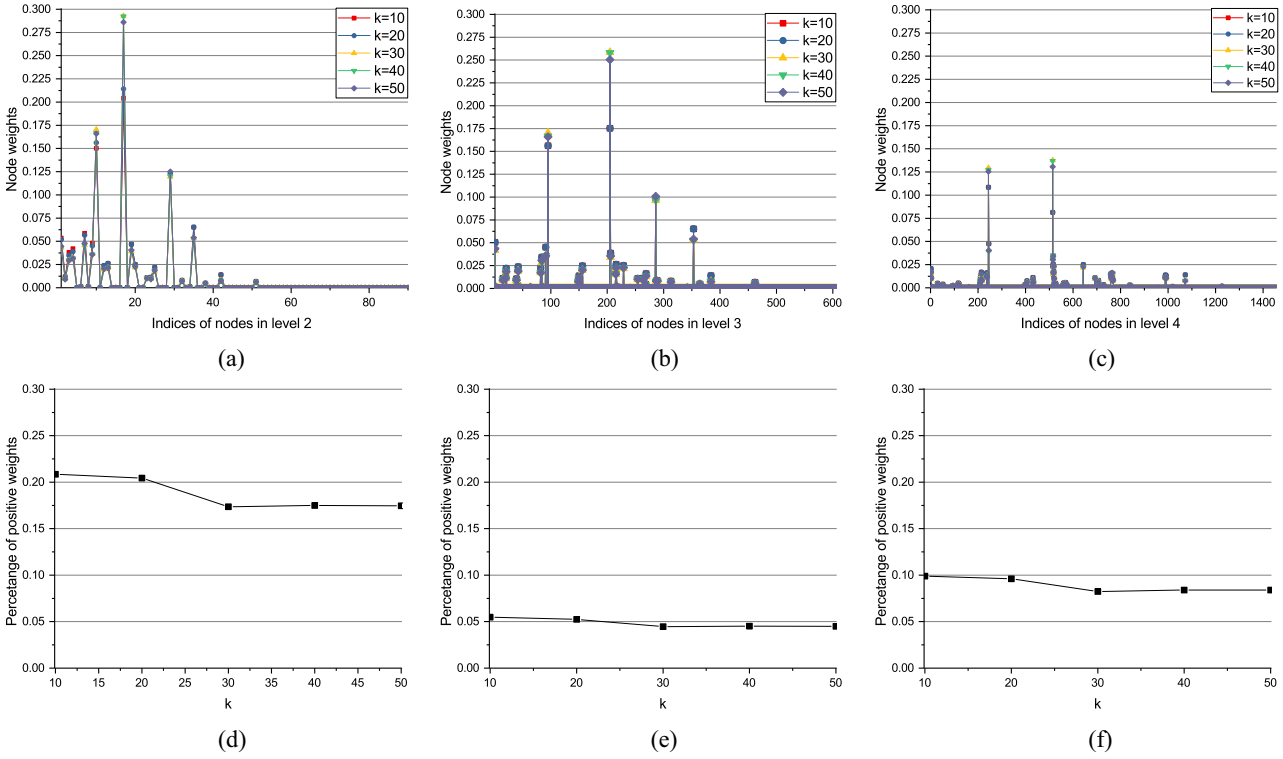

Fig. 9. Node weights Ω versus k on D_4. (a) $\{\omega_v | v \in N^2(\Phi)\}$ versus k. (b) $\{\omega_v | v \in N^3(\Phi)\}$ versus k. (c) $\{\omega_v | v \in N^4(\Phi)\}$ versus k. (d) Percentages of positive $\{\omega_v | v \in N^2(\Phi)\}$ versus k. (e) Percentages of positive $\{\omega_v | v \in N^3(\Phi)\}$ versus k. (f) Percentages of positive $\{\omega_v | v \in N^4(\Phi)\}$ versus k.

$\mathcal{NLW}$ than η. In real-applications, we can perform hierarchy grid search to choose the proper θ, η, and k for better result.

D. Weights Analysis

The relationships between the average level weights and three parameters θ, η, and k on D_4 are shown in Fig. 6. From

TABLE III
SAMPLES OF MOST IMPORTANT PRODUCTS SELECTED FROM THE CLUSTERING RESULTS OF TSW ON D_4, WHERE σ IS A THRESHOLD

σ	Most important products
0.1	
0.05	
0.01	vegetables, apple
0.005	vegetables, apple, leather shoes, pork
0.001	vegetables, apple, leather shoes, pork, pistachio, biscuits
0.0001	vegetables, apple, leather shoes, pork, pistachio, biscuits, beef, bacon

these figures, we can see that the average $\mathcal{NLW}$ are mainly affected by θ, and it is nearly stable with the change of both η and k. As θ increases, the level weights become flatter.

The relationships between the node weights Ω and three parameters θ, η, and k are shown in Figs. 7–9. From these figures, we can observe that the node weights Ω are mainly affected by η and it is nearly stable with the change of θ and k. The ratio of positive values in Ω increases with the increase of η, which can be verified according to (30). We also observe that the node weight distributions in different levels are highly correlated. Specifically, the average weight decreases with the increase of the tree level. This can be verified according to Theorem 1 which indicates that the sum of node weights in different levels are equal, and the fact that the high level contains more nodes than low level. With the increase of the tree level, the number of positive weights decreases rapidly. In real-world applications, we can adjust the distribution of Ω by η.

VII. CONCLUSION

In this paper, we have presented a TSW algorithm for customer transaction data. In the new method, a customer's transaction records are compressed into a "Purchase Tree," and a PurTree subspace distance is proposed to measure the dissimilarity between two customers represented by two purchase trees. A set of tree level weights and a set of sparse tree node weights are introduced into the PurTree subspace distance and to be learned. An iterative optimization algorithm TSW is proposed to optimize the new clustering model, in which a local similarity matrix and the two types of weights are simultaneously learned. Experimental results on ten real-life data sets have demonstrated the superior performance of the new method.

However, TSW has a high computational complexity of $O(rkn^2)$, where n is the number of purchase trees, k is the number of neighbors, and r is the number of iterations. In future work, we will improve this method for larger scale transaction data. Moreover, we will include additional information into the tree nodes, e.g., cost, quantity, etc., to form attributed PurTree. In such case, we will study new distance and new clustering method for the attributed PurTree data.

REFERENCES

[1] R. J. Kuo, L. M. Ho, and C. M. Hu, "Integration of self-organizing feature map and *K*-means algorithm for market segmentation," *Comput. Oper. Res.*, vol. 29, no. 11, pp. 1475–1493, 2002.

[2] C.-Y. Tsai and C.-C. Chiu, "A purchase-based market segmentation methodology," *Expert Syst. Appl.*, vol. 27, no. 2, pp. 265–276, 2004.

[3] T.-C. Lu and K.-Y. Wu, "A transaction pattern analysis system based on neural network," *Expert Syst. Appl.*, vol. 36, no. 3, pp. 6091–6099, 2009.

[4] T. Xiong, S. Wang, A. Mayers, and E. Monga, "DHCC: Divisive hierarchical clustering of categorical data," *Data Min. Knowl. Disc.*, vol. 24, no. 1, pp. 103–135, 2012.

[5] K. Wang, C. Xu, and B. Liu, "Clustering transactions using large items," in *Proc. 8th Int. Conf. Inf. Knowl. Manag.*, ACM, 1999, pp. 483–490.

[6] Y. Xiao and M. H. Dunham, "Interactive clustering for transaction data," in *Data Warehousing and Knowledge Discovery*. Heidelberg, Germany: Springer, 2001, pp. 121–130. [Online]. Available: https://link.springer.com/chapter/10.1007%2F3-540-44801-2_13

[7] F.-M. Hsu, L.-P. Lu, and C.-M. Lin, "Segmenting customers by transaction data with concept hierarchy," *Expert Syst. Appl.*, vol. 39, no. 6, pp. 6221–6228, 2012.

[8] U. Von Luxburg, "A tutorial on spectral clustering," *Stat. Comput.*, vol. 17, no. 4, pp. 395–416, 2007.

[9] H.-P. Kriegel, P. Kröger, and A. Zimek, "Clustering high-dimensional data: A survey on subspace clustering, pattern-based clustering, and correlation clustering," *ACM Trans. Knowl. Discov. Data*, vol. 3, no. 1, pp. 1–58, 2009.

[10] X. Li, G. Cui, and Y. Dong, "Graph regularized non-negative low-rank matrix factorization for image clustering," *IEEE Trans. Cybern.*, vol. 47, no. 11, pp. 3840–3853, Nov. 2017.

[11] Y. Yang, Z. Ma, Y. Yang, F. Nie, and H. T. Shen, "Multitask spectral clustering by exploring intertask correlation," *IEEE Trans. Cybern.*, vol. 45, no. 5, pp. 1083–1094, May 2015.

[12] D. Huang, C.-D. Wang, and J.-H. Lai, "Locally weighted ensemble clustering," *IEEE Trans. Cybern.*, vol. 48, no. 5, pp. 1460–1473, May 2018.

[13] X. Chen, J. Z. Huang, and J. Luo, "PurTreeClust: A purchase tree clustering algorithm for large-scale customer transaction data," in *Proc. IEEE 32nd Int. Conf. Data Eng. (ICDE)*, May 2016, pp. 661–672.

[14] X. Chen *et al.*, "Purtreeclust: A clustering algorithm for customer segmentation from massive customer transaction data," *IEEE Trans. Knowl. Data Eng.*, vol. 30, no. 3, pp. 559–572, Mar. 2018.

[15] X. Chen, S. Peng, J. Z. Huang, F. Nie, and Y. Ming, "Local PurTree spectral clustering for massive customer transaction data," *IEEE Intell. Syst.*, vol. 32, no. 2, pp. 37–44, Mar./Apr. 2017.

[16] J. Z. Huang, M. K. Ng, H. Rong, and Z. Li, "Automated variable weighting in k-means type clustering," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 5, pp. 657–668, May 2005.

[17] C. Domeniconi *et al.*, "Locally adaptive metrics for clustering high dimensional data," *Data Min. Knowl. Disc.*, vol. 14, no. 1, pp. 63–97, 2007.

[18] L. Jing, M. K. Ng, and J. Z. Huang, "An entropy weighting *k*-means algorithm for subspace clustering of high-dimensional sparse data," *IEEE Trans. Knowl. Data Eng.*, vol. 19, no. 8, pp. 1026–1041, Aug. 2007.

[19] Z. Deng, K. Choi, F. Chung, and S. Wang, "Enhanced soft subspace clustering integrating within-cluster and between-cluster information," *Pattern Recognit.*, vol. 43, no. 3, pp. 767–781, 2010.

[20] X. Chen, X. Xu, J. Z. Huang, and Y. Ye, "TW-k-means: Automated two-level variable weighting clustering algorithm for multi-view data," *IEEE Trans. Knowl. Data Eng.*, vol. 25, no. 4, pp. 932–944, Apr. 2013.

[21] X. Chen, Y. Ye, X. Xu, and J. Z. Huang, "A feature group weighting method for subspace clustering of high-dimensional data," *Pattern Recognit.*, vol. 45, no. 1, pp. 434–446, 2012.

[22] X. Huang *et al.*, "DSKmeans: A new kmeans-type approach to discriminative subspace clustering," *Knowl. Based Syst.*, vol. 70, pp. 293–300, Nov. 2014.

[23] X. Chen, M. Yang, J. Z. Huang, and Z. Ming, "TWCC: Automated two-way subspace weighting partitional co-clustering," *Pattern Recognit.*, vol. 76, pp. 404–415, Apr. 2018.

[24] A. Beygelzimer, S. Kakade, and J. Langford, "Cover trees for nearest neighbor," in *Proc. 23rd Int. Conf. Mach. Learn.*, ACM, 2006, pp. 97–104.

[25] D. R. Karger and M. Ruhl, "Finding nearest neighbors in growth-restricted metrics," in *Proc. 34th Annu. ACM Symp. Theory Comput. (STOC)*, 2002, pp. 741–750.

[26] F. Nie, X. Wang, and H. Huang, "Clustering and projected clustering with adaptive neighbors," in *Proc. 20th ACM SIGKDD Int. Conf. Knowl. Disc. Data Min.*, ACM, 2014, pp. 977–986.

[27] F. R. Chung, *Spectral Graph Theory*, vol. 92. Providence, RI, USA: Amer. Math. Soc., 1997.

[28] B. Mohar, Y. Alavi, G. Chartrand, and O. Oellermann, "The Laplacian spectrum of graphs," in *Graph Theory Combinatorics Appl.*, vol. 2. New York, NY, USA: Wiley, 1991, pp. 871–898.
[29] S. P. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, U.K.: Cambridge univ. Press, 2004.
[30] A. Y. Ng, M. I. Jordan, and Y. Weiss, "On spectral clustering: Analysis and an algorithm," in *Proc. Adv. Neural Inf. Proc. Syst. 14*, 2002, pp. 849–856. [Online]. Available: http://papers.nips.cc/paper/2092-on-spectral-clustering-analysis-and-an-algorithm/bibtex
[31] L. Hagen and A. B. Kahng, "New spectral methods for ratio cut partitioning and clustering," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 11, no. 9, pp. 1074–1085, Sep. 1992.

Xiaojun Chen (M'16) received the Ph.D. degree from the Harbin Institute of Technology, Harbin, China, in 2011.

He is currently an Assistant Professor with the College of Computer Science and Software, Shenzhen University, Shenzhen, China. His current research interests include machine learning and data mining.

Wenya Sun is currently pursuing the M.A. degree with the College of Computer Science and Software, Shenzhen University, Shenzhen, China.

Her current research interests include clustering and feature selection.

Bo Wang is currently pursuing the M.A. degree with the College of Computer Science and Software, Shenzhen University, Shenzhen, China.

His current research interests include clustering and feature selection.

Zhihui Li received the B.S. degree from the Beijing University of Posts and Telecommunications, Beijing, China, in 2008. She is currently pursuing the Ph.D. degree with the School of Computer Science and Engineering, University of New South Wales, Sydney, NSW, Australia.

She has been a Research Assistant with the School of Computer Science and Technology, Shandong University, Jinan, China, since 2018. She was a Data Analyst with Beijing Etrol Technologies Company Ltd., Shenzhen, China, until 2017. Her current research interests include artificial intelligence, machine learning, and computer vision.

Xizhao Wang (M'01–SM'06–F'12) received the Ph.D. degree in computer science from the Harbin Institute of Technology, Shenzhen, China, in 1998.

From 1998 to 2001, he was with the Department of Computing, Hong Kong Polytechnic University, Hong Kong, as a Research Fellow. From 2001 to 2014, he served with Hebei University, Baoding, China, as a Professor and the Dean of the School of Mathematics and Computer Sciences. Since 2014, he has been a Professor with the Big Data Institute of Shenzhen University, Shenzhen. He has edited over 10 special issues and published three monographs, two textbooks, and over 200 peer-reviewed research papers. His current research interests include uncertainty modeling and machine learning for big data.

Dr. Wang was a recipient of the IEEE SMCS Outstanding Contribution Award in 2004 and the IEEE SMCS Best Associate Editor Award in 2006. He is the previous BoG Member of IEEE SMC Society, the Chair of IEEE SMC Technical Committee on Computational Intelligence, the Chief Editor of the *International Journal of Machine Learning and Cybernetics*, and an associate editors for a couple of journals in the related areas. He is the General Co-Chair of the 2002–2017 International Conferences on Machine Learning and Cybernetics, cosponsored by IEEE SMCS. He was a Distinguished Lecturer of the IEEE SMCS.

Yunming Ye received the Ph.D. degree in computer science from Shanghai Jiao Tong University, Shanghai, China.

He is a Professor with Shenzhen Graduate School, Harbin Institute of Technology, Shenzhen, China. His current research interests include data mining, text mining, and ensemble learning algorithms.