PurTreeClust: A Purchase Tree Clustering Algorithm for Large-scale Customer Transaction Data

Xiaojun Chen
College of Computer Science and Software
Shenzhen University
Shenzhen 518060, P.R. China.
Email: xjchen@szu.edu.cn

Joshua Zhexue Huang
College of Computer Science and Software
Shenzhen University
Shenzhen 518060, P.R. China.
Email: zx.huang@szu.edu.cn

Jun Luo
Ecosystem & Cloud Service
Lenovo Group Limited
SIAT, CAS, China
Email: jun.luo@siat.ac.cn

***Abstract*—Clustering of customer transaction data is usually an important procedure to analyze customer behaviors in retail and e-commerce companies. Note that products from companies are often organized as a product tree, in which the leaf nodes are goods to sell, and the internal nodes (except root node) could be multiple product categories. Based on this tree, we present to use a "personalized product tree", called purchase tree, to represent a customer's transaction data. The customer transaction data set can be represented as a set of purchase trees. We propose a PurTreeClust algorithm for clustering of large-scale customers from purchase trees. We define a new distance metric to effectively compute the distance between two purchase trees from the entire levels in the tree. A cover tree is then built for indexing the purchase tree data and we propose a leveled density estimation method for selecting initial cluster centers from a cover tree. PurTreeClust, a fast clustering method for clustering of large-scale purchase trees, is then presented. Last, we propose a gap statistic based method for estimating the number of clusters from the purchase tree clustering results. A series of experiments were conducted on ten large-scale transaction data sets which contain up to four million transaction records, and experimental results have verified the effectiveness and efficiency of the proposed method. We also compared our method with three clustering algorithms, e.g., spectral clustering, hierarchical agglomerative clustering and DBSCAN. The experimental results have demonstrated the superior performance of the proposed method.**

I. Introduction

In retail and e-commerce companies, the products are often organized according to multiple types of categories. For example, a retailer may classify each product according to the business unit, the category, the subcategory, the product description and the brand. In a Chinese chain supermarket, more than 100,000 products can be classified in nearly 100 business units (e.g. Drinks, Fruits, Office supplies), in more than 500 categories (e.g. Beers, Fruit juice), in more than 1500 subcategories (e.g. Beers with alcohol, Pure fruit juice) and in more than 5,000 brands (i.e. Tsingtao beer, Lakewood juice).

Transaction data is the information recorded from daily transactions of customers. A transaction is a sequence of products (items) bought by a customer in one basket. Clustering of customer transaction data is usually the first step towards the analysis of customer behavior in retail and e-commerce companies [1], [2]. It is used to categorize customers into different 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. Early works use general variables like customer demographics and lifestyles [3], but such works are doubtful because the general variables are difficult to collect and some collected variables might be invalid soon without revision [3], [4]. With the rapid increase of the collected customer behavior data, the new study turns to cluster customers from transaction data [4]–[6]. However, since a customer often buy a very small proportion of the products provided by a company, such method suffers from the curse of high-dimensionality [7]. Recently, Hsu et. el. proposed a customer clustering method based on concept hierarchy of items, which defines a similarity measure based on path length and path depth on the concept hierarchy, and used hierarchical clustering to segment customers [8]. However, the distance is computed based on transactions and the method suffers from the huge amount of transaction records. Moreover, the high computation complexity of hierarchical clustering hinders the using of their customer segmentation methods in real applications.

In this paper, we propose a new data representation method. We first build a **product tree**, where the leaf nodes usually denote the provided products and the internal nodes are multiple product categories. A product tree often consists of several levels and thousands of nodes, and the number of nodes increases rapidly from the top level to the bottom level. In transaction data, each product (item) bought by a customer corresponds to a leaf node in the product tree. To facilitate the analysis and visualization of customer's behavior, we build a "personalized product tree" for each customer, called **purchase tree**. The purchase tree can be built by aggregating all products in a customer's transactions, and pruning the product tree by only retaining the corresponding leaf nodes and all paths from root to leaf node. In general, it can be a subset of a product tree and the number of purchase trees will be far less than the number of transactions. In fact, since a customer often buy a very small proportion of goods indicated by the product tree, the purchase tree may contain a much smaller number of nodes than those in the product tree. For example, a retail chain supermarket may sell more than 30 thousands of different goods, but a customer might only buy less than 100 goods.

To cluster customers instead of transaction records, we propose a new clustering method for clustering customer purchase trees. We define a new distance metric to effectively compute the distance of two purchase trees from the entire levels of the product tree, and provide a fast clustering method

978-1-5090-2020-1/16/$31.00 © 2016 IEEE

for clustering of large-scale purchase tree data. The main contributions of this paper include:

- We propose a novel PurTree distance metric, which is a weighted combination of a leveled tree distance between two purchase trees on each level, and the leveled tree distance is proved strictly increasing with the increase of tree level.
- We propose to build a cover tree for indexing the purchase tree data set and a fast leveled density estimation method for selecting initial cluster centers from a cover tree.
- We propose PurTreeClust, a fast clustering method for clustering large-scale purchase tree data, which takes time at most $O((c^6 + k)n\log(n))$, where n is the number of purchase trees, k is the number of clusters and c is the expansion constant.
- We propose a gap statistic based method for estimating the number of clusters from the purchase tree clustering results.

We used ten real-life transaction data sets from a famous super market in China, which contains up to four million newest transaction records in 2015. A series of experiments were conducted to investigate the properties of the new distance metric and the performance of the PurTreeClust algorithm. The experimental results have shown that the new algorithm is effective. We also compared our method with spectral clustering, hierarchical agglomerative clustering and DBSCAN algorithms. The experimental results have demonstrated the superior performance of our method.

The rest of this paper is organized as follows. Section II presents a brief review of related work. Notations and preliminaries are given in Section III. We present the PurTree distance metric and PurTreeClust algorithm in Section IV. The experimental results are reported in Sections V. Conclusions and future work are given in Section VI.

II. Related Work

Clustering of customers, also called customer segmentation, is one of the most critical elements in achieving successful modern marketing and customer relationship management [9], [10]. The early segmentation methods use general variables like customer demographics, lifestyle, attitude and psychology, because such variables are intuitive and easy to operate [11]. With the rapid increase of customer behavior data, the new study turns to use product specific variables such as items purchased [4], [5], [12], [13]. These methods often define distance on transaction records but such distance functions suffer from the curse of high-dimensionality [7]. Recently, Hsu et. el. proposed a customer clustering method based on concept hierarchy of items, which defines a similarity measure based on path length and path depth on the concept hierarchy [8]. However, the distance is also computed based on transaction records. Given a distance function, hierarchical clustering and genetic algorithm are often used for clustering [1], [4], [8]. However, such methods cannot handle large-scale transaction data due to their high computational complexity.

In the past decades, many clustering methods have been proposed, including hierarchical clustering [14], subspace clustering [15], co-clustering [15], density-based clustering [16] and spectral clustering [17]. Since the distance functions defined on transaction data are different from the traditional distance such as Euclidean distance, the commonly-used k-means clustering method cannot be used for clustering [1], [4], [8]. Given a similarity/distance matrix, hierarchical clustering, spectral clustering and density based clustering are three commonly-used clustering techniques.

Hierarchical clustering is a method of cluster analysis which seeks to build a hierarchy of clusters [14]. Two commonly-used strategies for hierarchical clustering are agglomerative and divisive. In hierarchical agglomerative clustering, each object starts in a cluster, and pairs of clusters are recursively merged to one cluster. In hierarchical divisive clustering, all objects start in one cluster, and a cluster is split recursively until it only contains one object. Hierarchical clustering often evolves merging or splitting subsets of objects rather than individual objects. Single link, average link, and complete link are three commonly-used distance functions to merge two clusters. Most hierarchical clustering algorithms have computational complexity at least $O(n^2)$ where n is the size of data. This high cost limits their applications in large-scale data. Although some improved hierarchical algorithms have computational complexity of $O(n\log(n))$ but they only work for numeric data [18].

Spectral clustering received a significant amount of attention in the last few years. Such methods make use of the spectrum (eigenvalues) of the similarity matrix of the data to perform dimensionality reduction, and then an iterative clustering algorithm is used to cluster the data with fewer dimensions [17], [19], [20].

Density based clustering estimates the density of objects in a data set and uncovers clusters from high-density areas. In density-based clustering, a cluster is a set of data objects spread over a contiguous region of high density objects, and objects in low-density regions are typically considered noise or outliers [16]. The typical density based clustering is DBSCAN, which finds core objects with high density and connects close core objects to form clusters [21]. Density-based clustering can find arbitrary shape clusters and handle noise data well, but it is slow due to neighborhood search for each data object, and faces a difficulty in setting the density threshold parameters properly.

III. Notations and Preliminaries

A. Notations

Let T be a tree with nodes $N(T)$ and edges $E(T) \in N(T) \times N(T)$. The root of T is denoted by $root(T)$. A node without children is a leaf, and otherwise an internal node. For an edge $\{v, w\} \in E(T)$, node v is the parent and w is the child, $P_w(T) = v$ and $w \in C_w(T)$. The descendants of v, the nodes in all paths reachable from v to a leaf node, is denoted as $des(v)$. The level of a node is defined by 1 + (the number of edges between the node and the root). $N^l(T)$ represents nodes in the l-th 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 and a leaf node.

B. Transaction Data

Let $R = \{r_1, ..., r_m\}$ be a set of m transaction records and $I = \{item_1, ..., item_y\}$ be a set of y items. A transaction record r_i is an itemset, represented as $(x_1, ..., x_z)$ where $x_j \in I$ for $1 \leq j \leq z$.

C. Product Tree

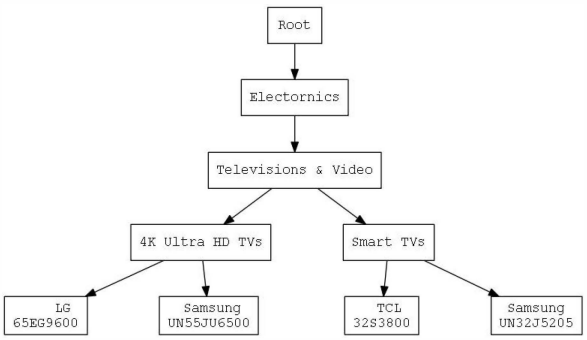


Fig. 1. A sample of product tree.

Let Ψ be a tree used to systematically organize the products sold by a company, in which the root node $root(\Psi)$ is an empty node "Root", each leaf node represents an product (item) and the internal nodes represent categories used to organize the products. In this paper, we assume that all leaf nodes in Ψ have equal depth, which means that all products are organized with the same numbers of categories. Level l ($2 \leq l \leq H(\Psi)$) represents the l-th category and the nodes in $N^l(\Psi)$ represent the items in the l-th category. Usually, $|N^l(\Psi)| < |N^{l+1}(\Psi)|$. Here, $I = N^{H(\Psi)+1}(\Psi)$, i.e., a transaction record r is a subset of the leaf nodes in Ψ. Figure 1 shows a sample of product tree which consists of three category levels. The category "Televisions & Video" consists of two sub-categories "4K Ultra HD" and "Smart TVs", each containing two products.

D. Purchase Tree

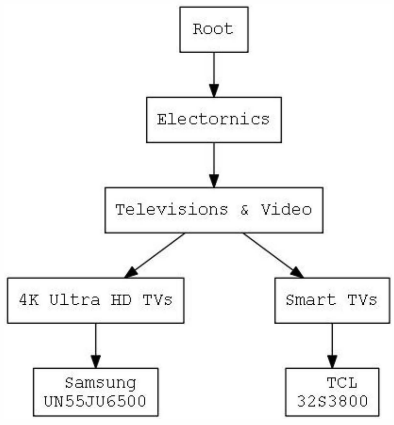


Fig. 2. A sample of purchase tree, built from the product tree in Figure 1.

A purchase tree φ is a subgraph of Ψ, i.e., $N(\varphi) \in N(\Psi)$, $E(\varphi) \in E(\Psi)$. Given any node $v \in N(\varphi)$, there must exist a leaf node $w \in N(\varphi)$ such that the path from $root(\varphi)$ to w also exists in Ψ. For each purchase tree φ, $H(\varphi) = H(\Psi)$. In this paper, we use $H(\Phi)$ to represent $H(\varphi)$. A purchase tree is used to illustrate the items bought by a customer, and it can be built from a customer's transaction data. Given a customer's m transaction records $R = \{r_1, ..., r_m\}$, the items in R can be aggregated as $I' \in I$. Then a purchase tree φ can be built from Ψ by only retaining the leaf nodes in I' and the internal nodes in the path from the root to each leaf node.

Figure 2 shows a sample of purchase tree, which is a subset of the product tree in Figure 1. A purchase tree often contains a very small subset of products in a product tree. For example, the product tree in Figure 1 consists of four products, but the purchase tree only contains two products. To evaluate the sparsity of a purchase tree, we define the sparsity of the l-th level in φ_i, with respect to the product tree Ψ, as

$$SP^l_\Psi(\varphi_i) = \frac{|N^l(\varphi_i)|}{|N^l(\Psi)|}. \quad (1)$$

Given a set of n purchase trees $\Phi = \{\varphi_1, ..., \varphi_n\}$, and a product tree Ψ, the average sparsity of the l-th level of Φ, denoted by $\overline{SP}^l_\Psi(\Phi)$, is defined as

$$\overline{SP}^l_\Psi(\Phi) = \frac{1}{n}\sum_{i=1}^{n} SP^l_\Psi(\varphi_i). \quad (2)$$

E. Cover Tree

Cover tree, first invented by Beygelzimer et. al., is a leveled tree data structure for fast nearest neighbor operations in metric spaces [22]. Let CT be a cover tree on a data set S and each level in CT be indexed by an integer scale l which decreases from top to bottom. Let CS_l denote the set of objects in S associated with the nodes at level l. Each object in S 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:

- **Nesting**: $CS_l \in CS_{l-1}$. If an object $p \in S$ appears in CS_l then every lower level in the tree has a node associated with p.
- **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.
- **Separation**: For two different objects $p, q \in CS_l$, $d(p, q) > 2^l$.

where d is a metric defined on S. Let c be the *expansion constant* of S [22], CT has the following properties:

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

The basic operation to build a cover tree is the insertion operation, which is shown in Algorithm 1. Here, Q_l is a subset of the objects at level l which may contain the new object p as a descendant. If S consists of n objects, the computational complexity of inserting an object is $O(c^6 \log(n))$ [22]. A cover tree can be built on a data set S by sequentially applying Algorithm 1 to insert each object in S.

Algorithm 1 Insert (object p, cover set Q_l, level l)

Ensure: **true** if insertion succeeds, **false** otherwise;
1: Set $Q = \{C_q : q \in Q_l\}$.
2: Set $Q_{l-1} = \{q \in Q : d(p,q) \leq 2^l\}$.
3: **if** $Q_{l-1} = \emptyset$ **then**
4: return **false**.
5: **else**
6: **if** Insert$(p, Q_{l-1}, l-1)$ fails **then**
7: Pick $q = \arg\min_{q \in Q_l} d(p,q)$.
8: **if** $d(q,p) = 0$ **then**
9: exit.
10: **else**
11: insert p as a child of q.
12: return **true**.
13: **end if**
14: **else**
15: return **true**.
16: **end if**
17: **end if**

IV. PurTreeClust clustering algorithm

In this section, we present the PurTreeClust algorithm for clustering large-scale purchase tree data. Note that the commonly-used distance, such Euclidian distance, cannot be used to characterize the purchase tree data. To address this, we propose a new distance metric which can effectively compute the difference between two purchase trees. By applying the proposed distance, we build a cover tree and propose a leveled density estimation method for selecting initial cluster centers. After that, we present a fast clustering algorithm for clustering large-scale purchase tree data. Finally, we propose a gap statistic based method for estimating the number of clusters from the purchase tree clustering results.

A. PurTree metric distance

Given a product tree Ψ and a set of n purchase trees $\Phi = \{\varphi_1, ..., \varphi_n\}$ where $\varphi_i \in \Psi$, let $H(\Phi)$ be the height of these purchase trees, $root(\varphi)$ be the empty root node of the purchase tree, $C_v(\varphi_i)$ be children nodes of v in φ_i, and $N^l(\varphi_i)$ be all nodes in the l-th level of φ_i. The Jaccard distance of two trees φ_i and φ_j on an internal node v is defined as

$$\delta_v(\varphi_i, \varphi_j) = \frac{|C_v(\varphi_i) \Delta C_v(\varphi_j)|}{|C_v(\varphi_i) \bigcup C_v(\varphi_j)|} \tag{3}$$

where $C_v(\varphi_i)\Delta C_v(\varphi_j) = C_v(\varphi_i) \bigcup C_v(\varphi_j) - C_v(\varphi_i) \bigcap C_v(\varphi_j)$ is the symmetric difference of two children nodes. $\delta_v(\varphi_i, \varphi_j)$ is bounded in $[0,1]$: $\delta_v(\varphi_i, \varphi_j) = 0$ if $C_v(\varphi_i) \bigcap C_v(\varphi_j) = C_v(\varphi_i) \bigcup C_v(\varphi_j)$; $\delta_v(\varphi_i, \varphi_j) = 1$ if $C_v(\varphi_i) \bigcap C_v(\varphi_j) = \emptyset$.

Definition 1. *The leveled tree distance between two purchase trees φ_i and φ_j on the l-th level is defined as*

$$d^l(\varphi_i, \varphi_j) = \sum_{v \in N^l(\varphi_i) \cup N^l(\varphi_j)} \beta_v \delta_v(\varphi_i, \varphi_j) \tag{4}$$

where $1 \leq l \leq H(\varphi)$, β_v is the weight for node v such that

$$\beta_v = \begin{cases} 1 & \text{if } v = root(\Phi) \\ \frac{\beta_w}{|C_w(\varphi_i) \bigcup C_w(\varphi_j)|} & \text{where } v \in C_w(\varphi_i) \bigcup C_w(\varphi_j) \end{cases} \tag{5}$$

Definition 2. *The PurTree distance between two purchase trees $\varphi_i, \varphi_j \in \Phi$ is computed as a weighted combination of the leveled tree distances in (4),*

$$d(\varphi_i, \varphi_j, \gamma) = \sum_{l=1}^{H(\Phi)} \omega_l d^l(\varphi_i, \varphi_j) \tag{6}$$

where ω_l is the l-th level weight. $\{\omega_1, ..., \omega_{H(\Phi)}\}$ is a geometric sequence with common ratio γ ($\gamma >= 0$) under constraints $\sum_{l=1}^{H(\Phi)} \omega_l = 1$, defined as

$$\omega_l = \begin{cases} \frac{1-\gamma}{1-\gamma^{H(\Phi)}} \gamma^{l-1} & \text{for } \gamma > 0 \text{ and } \gamma \neq 1 \\ \frac{1}{H(\Phi)} & \text{for } \gamma = 1 \\ 1 & \text{for } \gamma = 0 \text{ and } l = 1 \\ 0 & \text{for } \gamma = 0 \text{ and } 1 < l \leq H(\Phi) \end{cases} \tag{7}$$

The distance metric $d(\varphi_i, \varphi_j, \gamma)$ can be adjusted by setting different γ as follows:

- If $\gamma = 0$, $\omega_1 = 1$ and other weights are 0, thus only the top level L_1 contributes to the distance;
- If $0 < \gamma < 1$, $\omega_{l+1} = \gamma \omega_l < \omega_l$, thus a parent level contribute more to the distance than its child level;
- If $\gamma = 1$, $\omega_l = \frac{1}{H(\Phi)}$, thus all levels contribute equally to the distance;
- If $\gamma > 1$, $\omega_{l+1} = \gamma \omega_l > \omega_l$, thus a child level contribute more to the distance than its parent level;
- If $\gamma \to \infty$, $\omega_{H(\Phi)} = 1$ and other weights are 0, thus only the bottom level $L_{H(\Phi)}$ contributes to the distance.

$d(\varphi_i, \varphi_j, \gamma)$ can be simplified as $d(\varphi_i, \varphi_j)$ if we do not care about the value of γ.

Lemma 1. *Given n purchase trees $\Phi = \{\varphi_1, ..., \varphi_n\}$, for two purchase trees φ_i, φ_j in Φ and $1 \leq l \leq H(\Phi)$, $d^l(\varphi_i, \varphi_j) \in [0,1]$.*

Proof:

Since $\delta_v(\varphi_i, \varphi_j)$ lies in $[0,1]$, we can easily verify that $d^l(\varphi_i, \varphi_j) \geq 0$ and

$$\begin{aligned} d^l(\varphi_i, \varphi_j) &= \sum_{v \in N^l(\varphi_i) \cup N^l(\varphi_j)} \beta_v \delta_v(\varphi_i, \varphi_j) \\ &\leq \sum_{v \in N^l(\varphi_i) \cup N^l(\varphi_j)} \beta_v \\ &= 1 \end{aligned}$$

This completes the proof. ■

Form the above lemma, we can easily prove the following theorem:

Theorem 1. *Given n purchase trees $\Phi = \{\varphi_1, ..., \varphi_n\}$, for two purchase trees φ_i and $\varphi_j \in \Phi$, $d(\varphi_i, \varphi_j, \gamma) \in [0, 1]$.*

Lemma 2. *If $1 \leq l < H(\Phi)$, $d^{l+1}(\varphi_i, \varphi_j) \geq d^l(\varphi_i, \varphi_j)$.*

Proof:

We denote $C_v(\varphi_i)\Delta C_v(\varphi_j)$ as $\triangle^v$ for node v, and $C_v(\varphi_i) \cap C_v(\varphi_j)$ as $\cap^v$, then $\delta_v(\varphi_i, \varphi_j)$ can be rewritten as

$$\delta_v(\varphi_i, \varphi_j) = \frac{\mid \triangle^v \mid}{\mid \triangle^v \mid + \mid \cap^v \mid} \tag{8}$$

The sum of the distances on v's children nodes is

$$\begin{aligned} \sum_{w \in C_v(\varphi_i) \cup C_v(\varphi_j)} \delta_w(\varphi_i, \varphi_j) = & \sum_{w \in \triangle^v} \delta_w(\varphi_i, \varphi_j) \\ & + \sum_{w \in \cap^v} \delta_w(\varphi_i, \varphi_j) \end{aligned} \tag{9}$$

For $w \in \triangle^v$, we have $C_w(\varphi_i) \cap C_w(\varphi_j) = \emptyset$, thus $\delta_w(\varphi_i, \varphi_j) = 1$. For $w \in \cap^v$, $\delta_w(\varphi_i, \varphi_j) \geq 0$. Then we give

$$\begin{aligned} \sum_{w \in C_v(\varphi_i) \cup C_v(\varphi_j)} \delta_w(\varphi_i, \varphi_j) &\geq \mid \triangle^v \mid \\ &= (\mid \triangle^v \mid + \mid \cap^v \mid)\delta_v(\varphi_i, \varphi_j) \end{aligned} \tag{10}$$

Introducing the node weight β defined in (5) into (10) gives

$$\sum_{w \in C_v(\varphi_i) \cup C_v(\varphi_j)} \beta_w \delta_w(\varphi_i, \varphi_j) \geq \beta_v \delta_v(\varphi_i, \varphi_j) \tag{11}$$

Then we have

$$\begin{aligned} d^{l+1}(\varphi_i, \varphi_j) &= \sum_{v \in N^{l+1}(\varphi_i) \cup N^{l+1}(\varphi_j)} \beta_v \delta_v(\varphi_i, \varphi_j) \\ &= \sum_{v \in N^l(\varphi_i) \cup N^l(\varphi_j)} \sum_{w \in C_v} \beta_w \delta_w(\varphi_i, \varphi_j) \\ &\geq \sum_{v \in N^l(\varphi_i) \cup N^l(\varphi_j)} \beta_v \delta_v(\varphi_i, \varphi_j) \\ &= d^l(\varphi_i, \varphi_j) \end{aligned} \tag{12}$$

■

From the above lemma, we can easily prove the following theorem:

Theorem 2. *Given n purchase trees $\Phi = \{\varphi_1, ..., \varphi_n\}$, for two purchase trees φ_i and $\varphi_j \in \Phi$. If $1 \leq l \leq t \leq H(\Phi)$, $d^l(\varphi_i, \varphi_j) \leq d^t(\varphi_i, \varphi_j)$.*

The above theorem ensures that the leveled PurTree distance increases with the increase of tree level, which satisfies our intuition.

Theorem 3. *$d(\varphi_i, \varphi_j)$ defined in (6) is a metric.*

Proof:

Since the leveled tree distance $\delta_v(\varphi_i, \varphi_j)$ is Jaccard distance, which is a metric on the collection of all finite sets [23], we can easily verify that $d(\varphi_i, \varphi_j) = d(\varphi_j, \varphi_i)$ (Symmetry), $d(\varphi_i, \varphi_i) = 0$ (Reflexivity) and $d(\varphi_i, \varphi_j) \geq 0$ for all φ_i and φ_j in Φ (Positivity).

Given three Purchase trees φ_i, φ_j and φ_t,

$$\begin{aligned} & d(\varphi_i, \varphi_j) + d(\varphi_j, \varphi_t) \\ &= \sum_{l=1}^{H(\Phi)} \omega_l \sum_{v \in N^l(\varphi_i) \cup N^l(\varphi_j)} \beta_v \delta_v(\varphi_i, \varphi_j) \\ &\quad + \sum_{l=1}^{H(\Phi)} \omega_l \sum_{v \in N^l(\varphi_j) \cup N^l(\varphi_t)} \beta_v \delta_v(\varphi_j, \varphi_t) \\ &\geq \sum_{l=1}^{H(\Phi)} \omega_l \sum_{v \in N^l(\varphi_i) \cup N^l(\varphi_t)} \beta_v \delta_v(\varphi_i, \varphi_t) \\ &= d(\varphi_i, \varphi_t) \end{aligned}$$

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

B. Density estimation with cover tree

Since the purchase tree distance defined in (6) is a metric, we can build a cover tree CT from a set of n purchase trees $\Phi = \{\varphi_1, ..., \varphi_n\}$, with Algorithm 2. We have shown in Theorem 1 that $d(\varphi_i, \varphi_j) \in [0, 1]$. From the separation invariant, we know that there is at most one purchase tree in level 0 and other trees lie behind level 0. Therefore, we insert each purchase tree into a cover tree by applying $Insert(\phi_i, CS_0, 0)$ where CS_0 is the root node of CT. Since the computational complexity of Algorithm 1 is $O(c^6 \log(n))$, the overall computational complexity of Algorithm 2 is $O(c^6 n \log(n))$.

Algorithm 2 BuildCoverTree (Purchase tree data $\Phi = \{\phi_1, ..., \phi_n\}$)

Ensure: A cover tree CT built on Φ.

```
for i = 1; i <= n; i++ do
    Insert(φ_i, CS_0, 0).
end for
```

With the cover tree CT, we want to find k purchase trees as initial cluster centers for clustering. To be able to separate these trees and form compact clusters, the k initial cluster centers should be dense and far apart from each other. If we simply compute the density for each object and select k densest objects as initial cluster centers, they may be too close to form separate clusters. Recall the separation invariant, for two objects p and q in CS_l, $d(p, q) > 2^l$, where 2^l can be consider as the distance threshold for CS_l and it increases with the increase of the tree level l. Therefore, we can select the purchase trees in level $l = \arg\max_l |CS_l| > k$ as candidate cluster centers. The next step is to estimate the density of the objects in CS_l and select k densest objects from CS_l. We define a cover tree based density as:

Definition 3. *Given a cover tree CT built on Φ with a metric d, the leveled density of an object $p \in CS_l$ is $den^l_{CT}(p) = |\{q \in \Phi \backslash \{p\} : d(p,q) \leq 2^l\}|$.*

The leveled density estimation algorithm is shown in Algorithm 3. Given an object $P \in CS_l$, to find all objects q in Φ such that $d(p,q) \geq 2^l$, we descend through the tree level by level, keeping track of a subset $Q_i \in CS_i$ of nodes that may contain q as a descendant. The algorithm iteratively constructs Q_{i-1} by expanding Q_i to its children in CS_{i-1} and throwing away any child q that cannot satisfy $d(p,q) \leq 2^l$. For any object $q \in \Phi$, $des(q)$ can be computed during the building tree procedure thus no more time is required during the density estimation.

Algorithm 3 EstimateDensity (cover tree CT, object $p \in CS_l$, level l)

Ensure: $den^l_{CT}(p)$
1: $den^l_{CT}(p) = 0$.
2: Set $Q_l = CS_l$.
3: **for** $i = l; i < -\infty; i--$ **do**
4: $Q = \{C_q : q \in Q_i\}$.
5: Set $\Lambda_{i-1} = \{q \in Q : d(p,q) \leq 2^l - 2^{i+1}\}$.
6: **for** each $q \in \Lambda_{i-1}$ **do**
7: $den^l_{CT}(p) += |des(q)|$.
8: **end for**
9: $Q_{i-1} = \{q \in Q : d(p,q) \in (2^l - 2^{i+1}, 2^l + 2^{i+1})\}$.
10: $den^l_{CT}(p) += |\{q \in Q - Q_{i-1} : d(p,q) \leq 2^l|\}$.
11: **end for**
12: $den^l_{CT}(p) += |\{q \in Q_{-\infty} : d(p,q) \leq 2^l\}|$.
13: **return** $den^l_{CT}(p)$.

Theorem 4. *Given a cover tree CT built on Φ with a metric d, for any object $p \in CS_l$, Algorithm 3 returns $den^l(p)$ defined in Definition 3.*

Proof:

For any $q \in CS_i$, the distance between q and any descendant q' is bounded by $d(q,q') < \sum_{j=i}^{-\infty} 2^j = 2^{i+1}$. For any $q \in Q_i$, Algorithm 3 checks $d(p,q)$ and processes C_q as follows:

- If $d(p,q) \leq 2^l - 2^{i+1}$, the distance between p and $q' \in des(q)$ is bounded by $d(p,q') \leq d(p,q) + d(q,q') < 2^l$, i.e., $q \cup des(q) \in den^l_{CT}(p)$;
- If $d(p,q) \geq 2^l + 2^{i+1}$, the distance between p and $q' \in des(q)$ is bounded by $d(p,q') \geq d(p,q) - d(q,q') > 2^l$, then $q \cup des(q) \notin den^l_{CT}(p)$;
- If $d(p,q) \in (2^l - 2^{i+1}, 2^l + 2^{i+1})$, the distance between p and $q' \in C_q$ is bounded by $d(p,q') \in (2^l - 2^{i+2}, 2^l + 2^{i+2})$. Then C_q will be added into Q_{i-1}.

Therefore, Algorithm 3 can never throw out a grandparent $q' \in des(p)$. ■

In Step 4, the number of children encountered is at most $O(c^4|Q_i|)$. Step 5-10 takes time at most $O(c^4|Q_i|)$, and Step 12 requires $O(\max_i |Q_i|)$ time. Consequently, the running time is bounded by $O(c^4 \sum_{i=l}^{-\infty} |Q_i|)$. If in Step 5, $Q_{l-2} = \emptyset$, then Algorithm 3 takes time $O(1)$. In the worst case, if the last $Q_{-\infty} = \Phi \backslash \{p\}$, Algorithm 3 takes time at most $O(n \log(n))$.

We discuss $|Q_i|$ in the following. It can be verified that $|Q_l| = |Q_{l-1}| = |Q_{l-2}|$. In each iteration $i \leq l-2$,

$$\begin{aligned}\Lambda_{i-1} &= \{q \in Q : d(p,q) \leq 2^l - 2^{i+1}\} \\ &= B(p, 2^l - 2^{i+1}) \bigcap Q \\ &\subseteq B(p, 2^l - 2^{i+1}) \bigcap CS_{i-1}\end{aligned}$$

$$\begin{aligned}Q_{i-1} &= \{q \in Q : d(p,q) \in (2^l - 2^{i+1}, 2^l + 2^{i+1})\} \\ &= [B(p, 2^l + 2^{i+1}) - B(p, 2^l - 2^{i+1})] \bigcap Q \\ &\subseteq B(p, 2^l + 2^{i+1}) \bigcap CS_{i-1} - \Lambda_{i-1}\end{aligned}$$

Then we have

$$\begin{aligned}\frac{|Q_{i-1}|}{|CS_{i-1}|} &\leq \frac{|Q_{i-1}|}{|\Lambda_{i-1}| + |Q_{i-1}|} \\ &= 1 - \frac{|\Lambda_{i-1}|}{|\Lambda_{i-1}| + |Q_{i-1}|} \\ &= 1 - \frac{|B(p, 2^l - 2^{i+1}) \bigcap CS_{i-1}|}{|B(p, 2^l + 2^{i+1}) \bigcap CS_{i-1}|}\end{aligned}$$

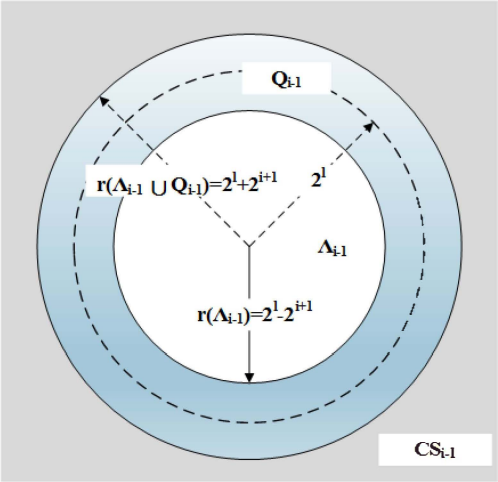
The relationship between Λ_{i-1} and CS_{i-1} is shown in Figure 3(a). We can see that with the decreasing of the tree level i, the ball of Λ_{i-1} will be closer to $B(p, 2^l)$ and Q_{i-1} will become smaller. Since it is very hard to bound $\frac{|B(p,2^l-2^{i+1}) \bigcap CS_{i-1}|}{|B(p,2^l+2^{i+1}) \bigcap CS_{i-1}|}$, we turn to compare the radii of two balls $B(p, 2^l - 2^{i+1})$ and $B(p, 2^l + 2^{i+1})$. Let $i' = l - i - 1$, and $r(B(.))$ be the radius of a ball $B(.)$, we define function $f(i')$ and draw it in Figure 3(b):

$$f(i') = \frac{r(B(p, 2^l - 2^{i+1}))}{r(B(p, 2^l + 2^{i+1}))} = 1 - \frac{2}{2^{i'} - 1}$$

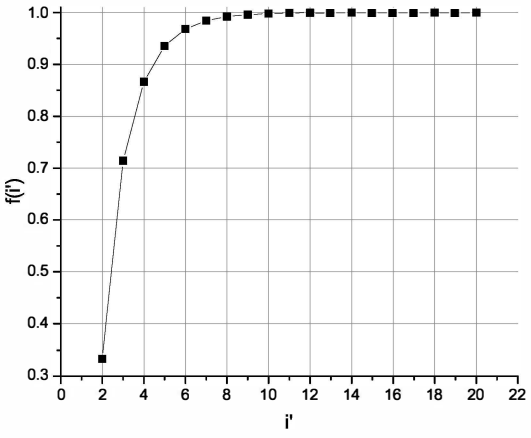
From Figure 3(b), we can see that $f(i')$ decreases rapidly with the increase of i'. For example, $r(B(p, 2^l - 2^{l-9}))) \approx 0.996 * r(B(p, 2^l + 2^{l-9}))$, i.e., $r(\Lambda_{l-10}) \approx 0.996 * r(\Lambda_{l-10} \bigcup Q_{l-10})$. In such situation, it may hold that $|Q_{l-10}| << |\Lambda_{l-10}|$, thus $|Q_{l-10}| << |CS_{l-10}|$. The following conclusion can be made based on the above analysis: as the tree level i decreases, since $|CS_{i-1}| \leq n$, although $|Q_{i-1}|$ may increase at the very start, it will decrease rapidly.

C. PurTreeClust algorithm

Given a cover tree CT built from n purchase trees, we want to cluster the n trees into k clusters. The basic method is to find k purchase trees as initial cluster centers, and then assign each customer to the nearest cluster center. We first select a proper level l where $|CS_l| > k$, estimate their densities with Algorithm 3, then the k initial cluster centers can be selected as the k densest trees from CS_l. If the cover tree is well distributed, i.e., the number of objects in cover set increases evenly with the decrease of the tree level, we can select a proper level l such that $|CS_l|$ is not too much bigger than k. However, in the worst case, $|CS_l|$ may be close to n. In such case, we can set Q as $2k$ objects in CS_l with the biggest



(a) Example of Λ_{i-1}, Q_{i-1} and CS_{i-1}.



(b) Figure of function $f(i')$.

Fig. 3. The relationship between Λ_{i-1}, Q_{i-1} and CS_{i-1}.

Algorithm 4 PurTreeClust (cover tree CT, no. of clusters k)

Ensure: Clustering result

1: // Find k cluster centers.
2: $Q_0 = CS_0$, where CS_0 is the root level of CT.
3: **for** $i = 0$; $i <= n$ and $|Q_i| < k$; $i--$ **do**
4: $Q_{i-1} = \{C_q : q \in Q_i\}$.
5: **end for**
6: Set $Q = \{q \in Q_i : \forall q' \notin Q, |des(q)| > |des(q')|, |Q| \geq 2k\}$
7: **for** each object $q \in Q$ **do**
8: $den^i(q)$ =EstimateDensity(CT, q, i).
9: **end for**
10: Set $U = \{q \in Q : \forall q' \notin U, den^i(q) > den^i(q'), |U| = k\}$
11: // Clustering.
12: **for** $i = 1; i <= n; i++$ **do**
13: $P_l = \arg\min_l d(\varphi_l, \varphi_i), \varphi_l \in U$.
14: **end for**
15: **return** P.

value of descendants. The details of the procedure is shown in Algorithm 4.

Given a set of n purchase trees, a cover tree CT can be built with Algorithm 2, which takes time at most $O(c^6 n \log(n))$. Then Algorithm 4 will return an expected clustering result with given k, which takes time at most $O(kn\log(n))$, so the overall procedure takes time at most $O(c^6 n\log(n) + kn\log(n))$, implying a bound of $O((c^6 + k)n\log(n))$. Such method suits well for changing business requirements. Once a cover tree is built, we can quickly cluster the data into different numbers of clusters. Moreover, a new customer can be directly inserted into the built cover tree without rebuilding the whole tree.

D. Choosing the number of clusters

Gap statistic is a popular method for choosing the number of clusters [24]. It computes the change in within-cluster dispersion with that expected under an appropriate reference null distribution. Given a purchase tree data set Φ, let $\log(W(k))$ be the within-cluster dispersion with k clusters computed as

$$\log(W(k)) = \log\{\sum_{l=1}^{k} \frac{1}{2|P_l|} \sum_{i,j \in P_l} d(\varphi_i, \varphi_j)\} \qquad (13)$$

where $\{P_1, ..., P_k\}$ is the clustering result of k partitions.

Let $\Upsilon = \{\Phi'_1, ..., \Phi'_B\}$ be a set of B sample data sets generated from a null reference distribution, which is assumed to contain a single component. We can cluster these data sets and compute $\{\log(W_1(k), ..., \log(W_B(k))\}$ from the clustering results. The gap statistic is then computed as the difference between the expectation of $\log(W(k))$ on reference data sets and $\log(W(k))$:

$$Gap(k) = \frac{1}{B}\sum_{j=1}^{B} \log(W_j(k)) - \log(W(k)) \qquad (14)$$

If the data consists of k^* well-separated clusters, we expect $\log(W(k))$ to decrease slowly when $k << k^*$ or $k >> k^*$; while decreasing slightly faster when k is close to k^*. In real applications, we can predefine a set of k, compute a set of $Gap(k)$ and draw them in a figure. Then the optimal number of clusters can be selected as the value of k for which $\log(W(k))$ falls the farthest below this reference curve, i.e., the biggest value of $Gap(k)$.

Given a set of n purchase trees $\Phi = \{\varphi_1, ..., \varphi_n\}$ and a set of numbers of clusters $K = \{k_1, ..., k_t\}$, the algorithm to compute t gap statistic values is shown in Algorithm 5. We first cluster Φ with different numbers of clusters and compute $\{\log(W(1)), ..., \log(W(t))\}$. To generate B reference data sets, we first compute the average sparsity of n purchase trees $\overline{SP}_\Psi(\Phi)$ and sample B purchase tree data sets $\{ST_1, ..., ST_B\}$. Each purchase tree in each reference data set consists of m sample purchase trees, whose nodes are uniformly sampled from Ψ, starting from level 2 down to level $H(\Psi)$+1. In the l-th level, $|N^l(\Psi)| * \overline{SP}^l_\Psi(\Phi)$ nodes are uniformly sampled from $\{v \in N^l(\Psi) : P_v \in N^{l-1}(\Psi)\}$.

Algorithm 5 Gap (product tree Ψ, n purchase trees $\Phi = \{\varphi_1, ..., \varphi_n\}$, a set of t numbers of clusters $K = \{k_1, ..., k_t\}$, number of copies B)

Ensure: A set of t gap statistic results

// Build a cover tree
$CT = BuildCoverTree(\Phi)$.
for $i = 1; i <= t; i++$ **do**
  $P_i = PurTreeClust(CT, k_i)$.
  Compute $\log(W(k_i))$ from P_i.
end for
// Generate B sample purchase trees.
Compute the average sparsity of n purchase trees $\overline{SP}_\Psi(\Phi) = \{\overline{SP}^1_\Psi(\Phi), ..., \overline{SP}^{H(\Psi)}_\Psi(\Phi)\}$.
for $j = 1; j <= B; j++$ **do**
  for $i = 1; i <= m; i++$ **do**
    $N^1(ST_{ij}) = \{$"*Root*"$\}$
    for $l = 2; j <= H(\Psi), l++$ **do**
      Uniformly sample $|N^l(\Psi)| * \overline{SP}^l_\Psi(\Phi)$ nodes from $\{v \in N^l(\Psi) : P_v \in N^{l-1}(\Psi)\}$ to form $N^l(ST_{ij})$.
    end for
  end for
  $SCT_j = BuildCoverTree(\{ST_{j1}, ..., ST_{jm}\})$.
end for
// compute gap statistic.
for $i = 1; i <= t; i++$ **do**
  for $j = 1; j <= B; j++$ **do**
    $P_i = PurTreeClust(SCT_j, k_i)$.
    Compute $\log(W_j(k_i))$ from P_i.
  end for
  $Gap(k_i) = (1/B)\sum_{j=1}^{B} \log(W_j(k_i)) - \log(W(k_i))$
end for
return $\{Gap(k_1), ..., Gap(k_t)\}$.

V. Experimental results and analysis

In this section, we present the experiments on real-life customer transaction data sets to investigate the effectiveness and scalability of the PurTreeClust algorithm.

A. Experiment setup

The customer transaction data set, from a big super market in China, contains up to four million newest transaction records in 2015, which consists of more than 40 thousand customers. The numbers of nodes in its product tree are 89, 579, 1,508, 5,489 and 22,666 from level L_1 to L_5, respectively. We sampled different numbers of customers to generate 10 data sets and computed their average sparsity, and the results are shown in Table I. We can see that the sparsity decrease rapidly from L_1 to L_5. This indicates that most customers only bought a very small proportion of the total products.

Hierarchical agglomerative clustering (HAC) [25], spectral clustering (SPEC) [1] and DBSCAN [21] were used for comparisons.

B. Experiments on PurTree distance

We have shown in Theorem 2 that the leveled tree distance increases with the increase of tree level. To verify this theorem, we computed the leveled tree distance of pair objects on D_4 and drew some samples in Figure 4. We can see that the leveled tree distances on all pairs strictly increased with the increase of tree level. Distances on L_1 mainly fell in the range of $(0.3, 1]$, and distances on L_4 mainly fell in the range of $[0.9, 1]$.

[1] http://www.luigidragone.com/software/spectral-clusterer-for-weka/

TABLE I. Characteristics and average sparsity of ten customer transaction data sets.

Data sets	Size	$SP^1_\Psi(D)$	$SP^2_\Psi(D)$	$SP^3_\Psi(D)$	$SP^4_\Psi(D)$	$SP^5_\Psi(D)$
D_1	215	1.539%	0.443%	0.253%	0.080%	0.029%
D_2	387	1.548%	0.455%	0.268%	0.087%	0.034%
D_3	698	1.410%	0.397%	0.226%	0.070%	0.026%
D_4	1258	0.564%	0.160%	0.088%	0.028%	0.010%
D_5	2265	0.561%	0.156%	0.084%	0.026%	0.009%
D_6	4078	0.623%	0.176%	0.097%	0.031%	0.011%
D_7	7342	0.569%	0.160%	0.090%	0.028%	0.010%
D_8	13216	0.286%	0.079%	0.044%	0.014%	0.005%
D_9	23790	0.29%	0.081%	0.045%	0.014%	0.005%
D_{10}	42823	0.286%	0.080%	0.045%	0.014%	0.005%

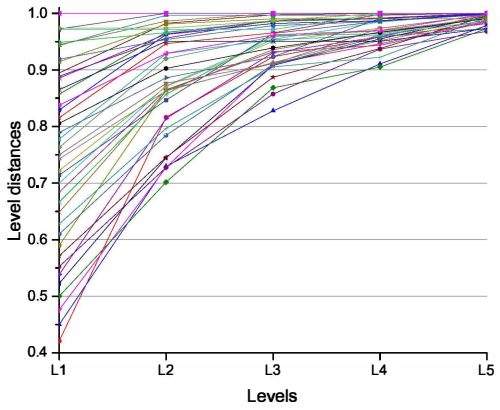
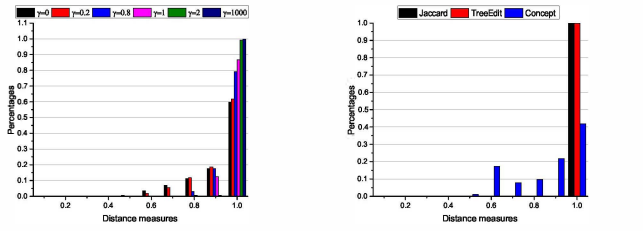


Fig. 4. Samples of the leveled tree distance.



(a) Distance distributions by the PurTree distance.

(b) Distance distributions by the Jaccard distance, TreeEdit distance and Concept dissimilarity.

Fig. 5. Comparison of distance distributions by four distance functions.

In the PurTree distance defined in (6), a positive distribution parameter γ is introduced to adjust the contributions of the purchase tree's different levels to the final distance. To investigate the effect of γ on the PurTree distance, we set $\gamma = \{0, 0.2, 0.8, 1.0, 2.0, 1000\}$ and computed PurTree distance on pair of objects on D_4. The distribution of the distances is drawn in Figure 5(a). From this figure, we can see that as γ increased, the distance between two purchase

trees became closer to 1, thus the objects being far apart from each other. As γ decreased, the distances fell more evenly in $[0, 1]$. If γ is very big, e.g., 1000, the distances between objects will be too big to find compact clusters. But if γ is too small, e.g., 1000, the distances between objects may be too small thus they are too close to be separated. Therefore, we have to set γ in a reasonable range according to the domain knowledge.

We also computed the Jaccard distance (only compute distances on leaf nodes of the purchase tree), tree edit distance [26] and concept dissimilarity [8] on pairs of objects on D_4. The distributions of the distances are drawn in Figure 5(b). From this figure, we can see that more than 99% of the distances by the Jaccard distance and tree edit distance fell in [0.9,1.0], from which it is difficult to recover cluster structure. Compared with other two distances, the concept dissimilarity produced more evenly distributed distances. But it is difficult to set three parameters properly. Compared with the results in Figure 5(b), the results produced by the PurTree distance are more explicable. Moreover, we have proven that the PurTree distance is a metric. Therefore, many transaction data clustering algorithms can benefit from this distance.

C. Experiments on the PurTreeClust algorithm

With the cover trees built in the last experiment, we selected 36 integers of k from 5 to 40 to run the PurTreeClust algorithm and computed the gap statistic values with Algorithm 5. The results are drawn in Figure 6. We can see that both $E[\log(W(k))]$ and $\log(W(k))$ dropped slowly with the increase of k on all cover trees. With small γ, $\log(W(k))$ decreased faster than $E[\log(W(k))]$ with small k and then the difference between them became stable. With big γ, $\log(W(k))$ got close to $E[\log(W(k))]$ for almost all k. The gap results are drawn in Figure 7. We can see that the gap values were close to 0 when $\gamma = \{0.8, 1, 2, 1000\}$, which indicates that no cluster structure can be found with the four parameters. Recall the results in Figure 5(a), the distance between two purchase trees was too big when $\gamma = \{0.8, 1, 2, 1000\}$. Therefore, the customers were too far apart from each other to uncover compact cluster structure. $Gap(k)$ reached its largest value when $k = 9$ for $\gamma = 0$ and $k = 14$ for $\gamma = 0.2$. We selected $\gamma = 0.2$ and $k = 14$ for the following results because the differences in all levels were considered when $\gamma = 0.2$.

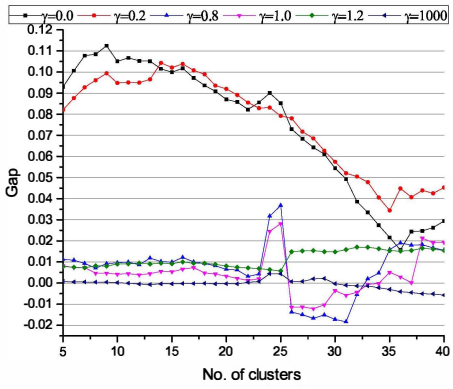
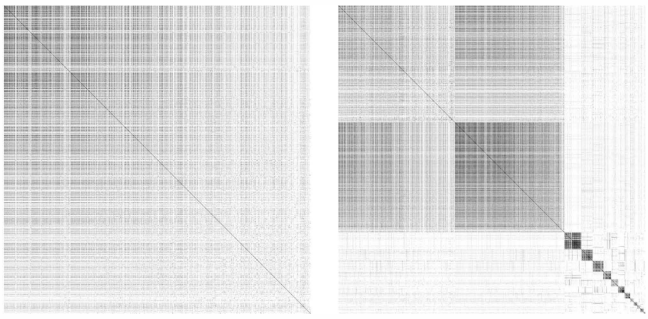


Fig. 7. Gap results of six clustering results by PurTreeClust on D_4.

To visually investigate the effectiveness of the clustering result by the PurTreeClust algorithm, we first computed the paired distances between trees in D_4 and drew the distance matrix in Figure 8(a), in which each row and column represent a purchase tree in the same order. The darker color represents lower distance and lighter color represents higher distance. From this figure, we cannot see clear cluster structure. To check whether clear cluster structure can be found from the clustering result by the PurTreeClust algorithm, we arranged the order of purchase trees in D_4 such that the trees in the same cluster were together, according to the clustering result with $\gamma = 0.2$ and $k = 14$. The result was drawn in Figure 8(b). From this figure, we can see clear cluster structure. Two big clusters consist of more than half of the whole trees in D_4 and other 12 clusters were small clusters.



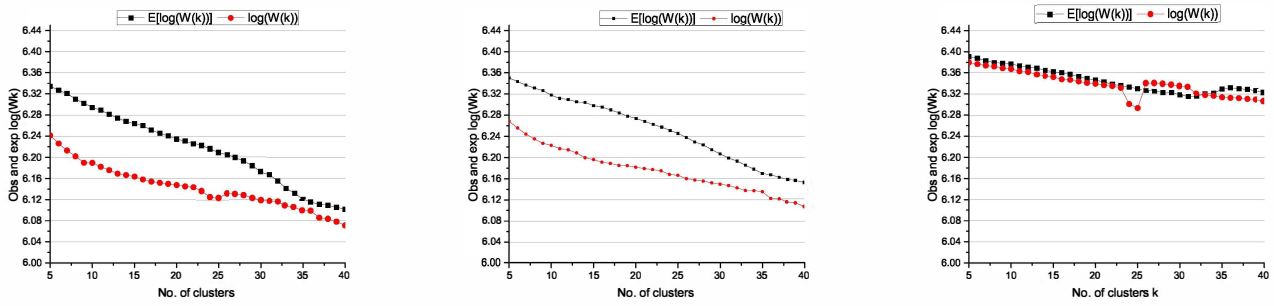
(a) Original PurTree distance matrix with $\gamma = 0.2$ on D_4.

(b) Clustering results by the PurTreeClust algorithm with $\gamma = 0.2$ and $k = 14$ on D_4.

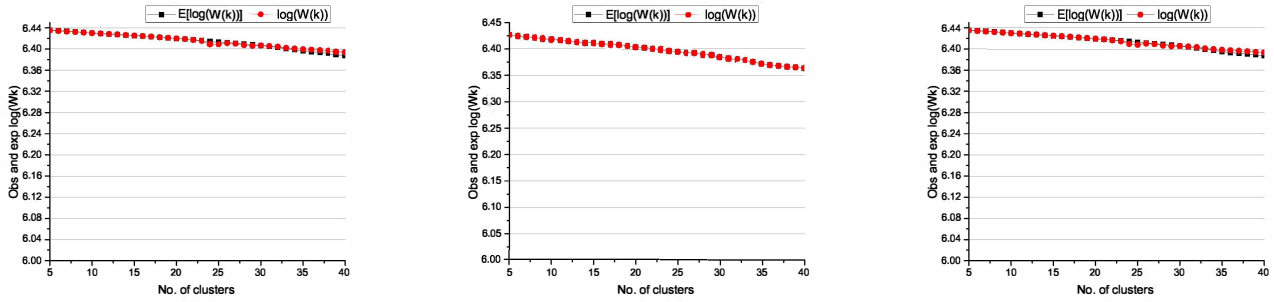
Fig. 8. Distance matrix on D_4.

To further investigate the clustering result by the PurTreeClust algorithm, we computed the sparsity of 14 cluster centers from the clustering result by the PurTreeClust algorithm with $\gamma = 0.2$ and $k = 14$. The results are drawn in Figure 9(a). From this figure, we can see that the sparsity of 2 clusters are significantly higher than those of other clusters. Besides the 2 big clusters, other clusters were small clusters with low sparsity. We also computed the average sparsity of 14 clusters and drew the results in Figure 9(b). Compared with the results in Figure 9(a), we can see that the average sparsity of 2 big clusters were smaller than the sparsity of 2 cluster centers, but the average sparsity of other clusters were close to or even bigger than the sparsity of corresponding cluster centers. Finally, we merged all purchase trees in a cluster into a cluster tree and drew them in Figure 9(c). We can see that the sparsity of 2 big clusters were still higher than those of other clusters. Interestingly, though the results of other 12 clusters were similar in Figure 9(a) and 9(b), their results were very different in Figure 9(c). The above results revealed the diversity of the uncovered cluster structures, thus verified the effectiveness of the PurTreeClust algorithm.

We reported the time costs of the PurTreeClust algorithm in the last experiment and the results are drawn in Figure 10, in which the time costs with $k = 0$ are the time for building corresponding cover tree and the time costs with $k > 0$ are the time for building a cover tree plus the time for clustering. We can see that the time cost of building a cover tree increased with the increase of γ. This can be explained from the covering invariant of cover tree, since the distance between a parent

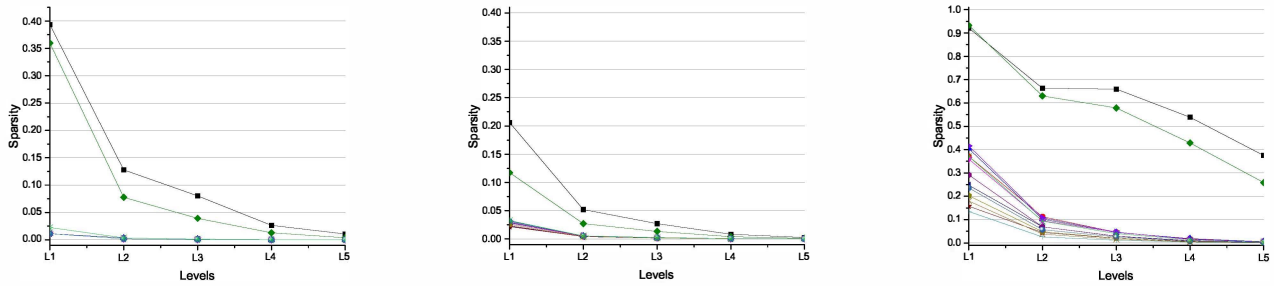


(a) Clustering results with $\gamma = 0$. (b) Clustering results with $\gamma = 0.2$. (c) Clustering results with $\gamma = 0.8$.



(d) Clustering results with $\gamma = 1$. (e) Clustering results with $\gamma = 2$. (f) Clustering results with $\gamma = 1000$.

Fig. 6. $E[\log(W(k))]$ and $\log(W(k))$ of clustering results by PurTreeClust on D_4.



(a) Sparsity of 14 cluster centers. (b) Average sparsity of trees in each of 14 clusters. (c) Sparsity of 14 cluster trees.

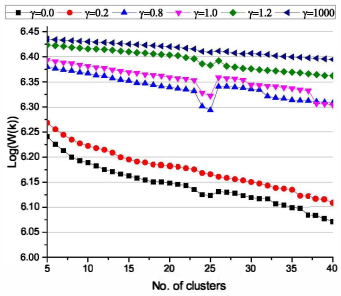
Fig. 9. Sparsity of clustering results.

node in the l-th and its child node must be less than 2^l. We know from the results in Figure 5 that as γ increased, the distance between two purchase trees became closer to 1, thus the cover tree contains less levels and more distance comparison are involved during building a cover tree. When $\gamma = \{0, 0.2\}$, the time costs of clustering had a sudden increase at $k = 26$, which was because new levels will be involved during the density estimation when $k \geq 26$. When $\gamma = \{0.8, 1, 2, 1000\}$, the time costs of the clustering increased slowly with the increase of k.

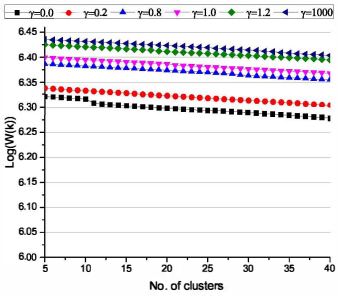
D. Comparison results and analysis

In this experiment, we compared the effectiveness of the PurTreeClust algorithm against SPEC, HAC with three linkages and DSCAN on D_4. We selected a set of $\gamma = \{0, 0.2, 0.8, 1, 2, 1000\}$ and k from 5 *to* 40 to run the five algorithms except for the PurTreeClust algorithm, and compared their results against the results by the PurTreeClust algorithm reported in the last section. The $\log(W(k))$ of six clustering algorithms were computed and the results are drawn in Figure 11. From these results, we can see that the PurTreeClust algorithm achieved the lowest values, thus outperformed other five clustering algorithms. Among three linkages for HAC, mean linkage achieved the best clustering results. SPEC and DBSCAN failed to uncover most specific number of clusters due to the difficulty on setting proper parameters.

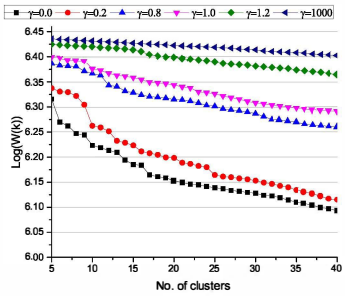
To compare the clustering results visually, we drew the clustering results by HAC with mean linkage and DBSCAN with $\gamma = 0.2$ and $k = 14$ in Figure 12, in a similar way



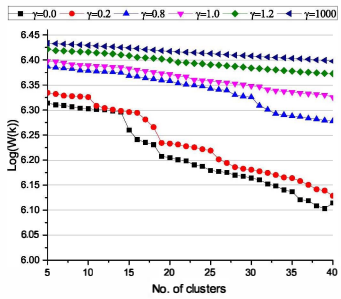
(a) Log($W(k)$) of the clustering results by PurTreeClust.



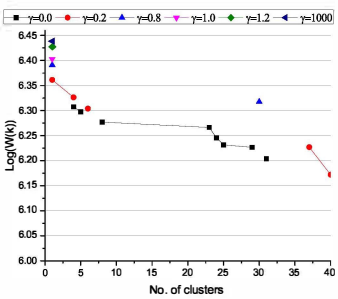
(b) Log($W(k)$) of the clustering results by HAC with single linkage.



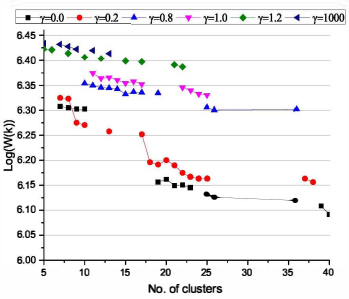
(c) Log($W(k)$) of the clustering results by HAC with mean linkage.



(d) Log($W(k)$) of the clustering results by HAC with complete linkage.



(e) Log($W(k)$) of the clustering results by SPEC.



(f) Log($W(k)$) of the clustering results by DBSCAN.

Fig. 11. Log($W(k)$) of the clustering results by six clustering algorithms on D_4.

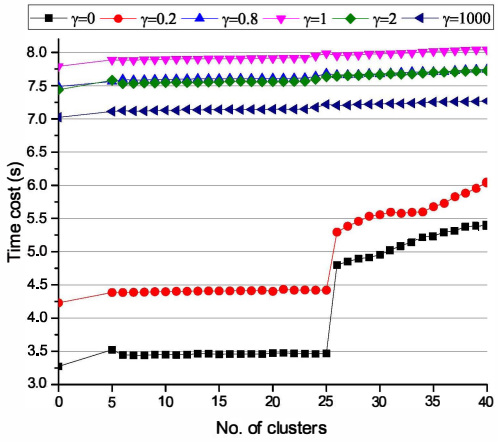
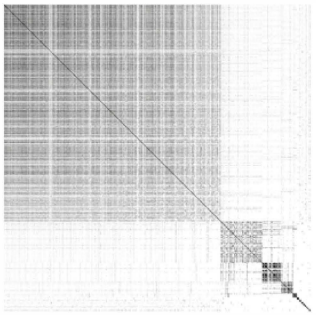
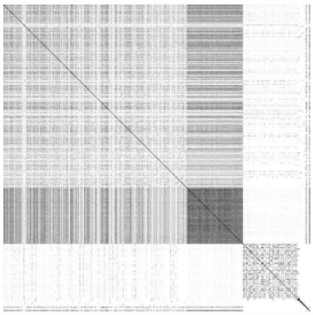


Fig. 10. The execution time of the PurTreeClust algorithm on D_4.



(a) Clustering results by the HAC clustering algorithm.



(b) Clustering results by the DBSCAN clustering algorithm.

Fig. 12. Clustering results by the HAC and DBSCAN clustering algorithms with the PurTree distance with $\gamma = 0.2$ and $k = 14$ on D_4.

to Figure 8. Compared with the results by the PurTreeClust algorithm in Figure 8, we can see that PurTreeClust uncovered more balanced clusters, e.g., the largest/smallest cluster had smaller/larger size than those uncovered by the other two algorithms. Thus the clustering result by the PurTreeClust algorithm had the lowest $\log(W(k))$.

E. Scalability comparison

We used the ten data sets in Table I to compare the scalability of the PurTreeClust algorithm against SPEC, HAC and DBSCAN. In this experiment, we selected a set of γ and k to run the three algorithms and computed the average time costs by three clustering algorithms. The results are drawn in Figure 13, in which SPEC, HAC and DBSCAN cannot finish clustering in accepted time on the last several data sets. Although the time costs by DBSCAN was lower than the time costs by the PurTreeClust algorithm at the first k, it soon exceeded the latter. In summary, the time costs by the PurTreeClust algorithm were much smaller than those by other three algorithms. This indicates that PurTreeClust scales better to large transaction data than the other three clustering algorithms.

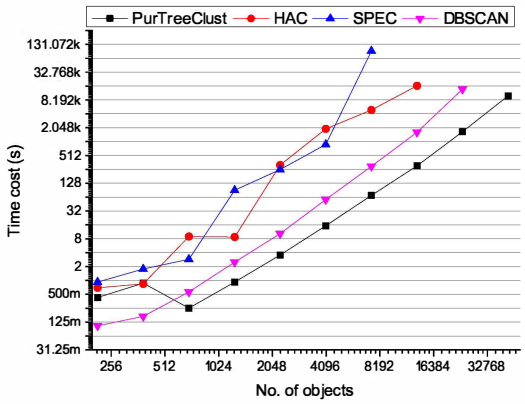


Fig. 13. The execution time of four clustering algorithms, e.g., PurTreeClust, HAC, SPEC and DBSCAN on 10 data sets listed in Table I.

VI. Conclusions

We have presented PurTreeClust, a purchase tree clustering algorithm for large-scale customer transaction data. In this method, a "personalized product tree" called "purchase tree" is built for each customer from the customer's transaction data. A new distance metric is defined to effectively compute the distance from two purchase trees. To cluster these purchase trees, we first build a cover tree for indexing the purchase tree data and propose a leveled density estimation method for selecting initial cluster centers from the cover tree. PurTreeClust, a fast clustering method for clustering of large-scale purchase trees, is then presented. A series of experiments were conducted on some large-scale transaction data sets which contain up to four million transaction records, and the experimental results have shown that the new distance metric is effective. The clustering algorithm can recover good cluster structure and it is scalable. We also compared our method with SPEC, HAC and DBSCAN algorithms and the experimental results have shown that PurTreeClust is more effective and scalable.

In the future work, we will introduce more features into purchase tree, such as monetary expense, etc. Furthermore, we will study clustering of mixture data, e.g., purchase tree and general variables. Since we propose a general distance metric for two subtrees, it could be used in clustering other objects embedded in a tree, e.g., species in phylogenetic tree. It will be interesting to apply our method in other fields.

Acknowledgment

This research was supported by NSFC under Grant no.61305059, 61473194 and 11271351, Natural Science Foundation of SZU (grant no. 201432) and PhD Start-up Fund of Natural Science Foundation of Guangdong Province (No. 2015A030310364).

References

[1] F. Giannotti, C. Gozzi, and G. Manco, "Clustering transactional data," in *Principles of Data Mining and Knowledge Discovery*. Springer, 2002, pp. 175–187.

[2] Y. Yang, X. Guan, and J. You, "Clope: a fast and effective clustering algorithm for transactional data," in *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2002, pp. 682–687.

[3] R. G. Drozdenko and P. D. Drake, *Optimal database marketing: strategy, development, and data mining*. Sage, 2002.

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

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

[6] V. L. Miguéis, A. S. Camanho, and J. F. e Cunha, "Customer data mining for lifestyle segmentation," *Expert Systems with Applications*, vol. 39, no. 10, pp. 9359–9366, 2012.

[7] D. Donoho, "High-dimensional data analysis: The curses and blessings of dimensionality (slides)," August 2000.

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

[9] A. Berson and S. J. Smith, *Building data mining applications for CRM*. McGraw-Hill, Inc., 2002.

[10] E. W. Ngai, L. Xiu, and D. C. Chau, "Application of data mining techniques in customer relationship management: A literature review and classification," *Expert systems with applications*, vol. 36, no. 2, pp. 2592–2602, 2009.

[11] R. Kuo, L. Ho, and C. M. Hu, "Integration of self-organizing feature map and k-means algorithm for market segmentation," *Computers & Operations Research*, vol. 29, no. 11, pp. 1475–1493, 2002.

[12] Y. Xiao and M. H. Dunham, "Interactive clustering for transaction data," in *Data Warehousing and Knowledge Discovery*. Springer, 2001, pp. 121–130.

[13] T. Xiong, S. Wang, A. Mayers, and E. Monga, "Dhcc: Divisive hierarchical clustering of categorical data," *Data Mining and Knowledge Discovery*, vol. 24, no. 1, pp. 103–135, 2012.

[14] R. Xu, D. Wunsch *et al.*, "Survey of clustering algorithms," *Neural Networks, IEEE Transactions on*, vol. 16, no. 3, pp. 645–678, 2005.

[15] 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.

[16] H.-P. Kriegel, P. Kröger, J. Sander, and A. Zimek, "Density-based clustering," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 1, no. 3, pp. 231–240, 2011.

[17] A. Y. Ng, M. I. Jordan, Y. Weiss *et al.*, "On spectral clustering: Analysis and an algorithm," *Advances in neural information processing systems*, vol. 2, pp. 849–856, 2002.

[18] T. Zhang, R. Ramakrishnan, and M. Livny, "Birch: an efficient data clustering method for very large databases," in *ACM SIGMOD Record*, vol. 25, no. 2. ACM, 1996, pp. 103–114.

[19] U. Von Luxburg, "A tutorial on spectral clustering," *Statistics and computing*, vol. 17, no. 4, pp. 395–416, 2007.

[20] K. Chaudhuri, F. C. Graham, and A. Tsiatas, "Spectral clustering of graphs with general degrees in the extended planted partition model." in *COLT*, vol. 23, 2012, pp. 35–1.

[21] J. Sander, M. Ester, H.-P. Kriegel, and X. Xu, "Density-based clustering in spatial databases: The algorithm gdbscan and its applications," *Data mining and knowledge discovery*, vol. 2, no. 2, pp. 169–194, 1998.

[22] A. Beygelzimer, S. Kakade, and J. Langford, "Cover trees for nearest neighbor," in *Proceedings of the 23rd international conference on Machine learning*. ACM, 2006, pp. 97–104.

[23] A. H. Lipkus, "A proof of the triangle inequality for the tanimoto distance," *Journal of Mathematical Chemistry*, vol. 26, no. 1-3, pp. 263–265, 1999.

[24] R. Tibshirani, G. Walther, and T. Hastie, "Estimating the number of clusters in a data set via the gap statistic," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 63, no. 2, pp. 411–423, 2001.

[25] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. Witten, "The weka data mining software: An update." 2013.

[26] M. Pawlik and N. Augsten, "Rted: a robust algorithm for the tree edit distance," *Proceedings of the VLDB Endowment*, vol. 5, no. 4, pp. 334–345, 2011.