Peer-reviewed version available at *Sustainability* **2018**, *10*, 2614; doi:10.3390/su10082614

Article

An Efficient Grid-based K-prototypes Algorithm for Sustainable Decision Making using Spatial Objects

Hong-Jun Jang [1], Byoungwook Kim [2], Jongwan Kim [3] and Soon-Young Jung [4,*]

[1] Department of Computer Science and Engineering, Korea University, Seoul, 02841, Korea; hongjunjang@korea.ac.kr
[2] Department of Computer Engineering, Dongguk University, Gyeongju, 38066, Korea; bwkim@dongguk.ac.kr
[3] Smith Liberal Arts College, Sahmyook University, Seoul, 01795, Korea; kimj@syu.ac.kr
[4] Department of Computer Science and Engineering, Korea University, Seoul, 02841, Korea; jsy@korea.ac.kr
* Correspondence: jsy@korea.ac.kr; Tel.: +82-2-3290-2394

Abstract: Data mining plays a critical role in the sustainable decision making. The k-prototypes algorithm is one of the best-known algorithm for clustering both numeric and categorical data. Despite this, however, clustering a large number of spatial object with mixed numeric and categorical attributes is still inefficient due to its high time complexity. In this paper, we propose an efficient grid-based k-prototypes algorithms, GK-prototypes, which achieves high performance for clustering spatial objects. The first proposed algorithm utilizes both maximum and minimum distance between cluster centers and a cell, which can remove unnecessary distance calculation. The second proposed algorithm as extensions of the first proposed algorithm utilizes spatial dependence that spatial data tend to be more similar as objects are closer. Each cell has a bitmap index which stores categorical values of all objects in the same cell for each attribute. This bitmap index can improve the performance in case that a categorical data is skewed. Our evaluation experiments showed that proposed algorithms can achieve better performance than the existing pruning technique in the k-prototypes algorithm.

Keywords: clustering; spatial data; grid-based k-prototypes; data mining; sustainability

1. Introduction

Sustainability is a concept for balancing environmental, economic and social dimensions with decision-making [23]. Data mining in sustainability is a very important issue since it sustainable decision making contributes to the transition to a sustainable society [24]. Especially, there is a growing interest in spatial data mining in making sustainable decisions in geographical environments and national land policies [25].

Recently, spatial data mining has become more and more important as spatial data collection is increasing due to technological developments such as geographic information system (GIS) and global positioning system (GPS) [26][27]. The main techniques of spatial data mining are spatial clustering [1], spatial classification [2], spatial association rule [3], and spatial characterization [4]. Spatial clustering is a technique used to classify data with high similar geographic and locational characteristics into the same group. It is an important component to discover hidden knowledge in a huge of spatial data [5]. Spatial clustering is used in the HotSpot detection which detects areas where specific events occur [6]. Hotspot detection is used in various fields such as crime analysis [7,8,9], fire analysis [10] and disease analysis [11,12,13,14].

Most spatial clustering studies have focused on efficiently finding groups for numeric data such as location information of spatial objects. However, many real-world spatial objects have categorical data, not just numeric data. Hence, if the categorical data affect spatial clustering results, the error value can be increased when the final cluster results are evaluated.

Peer-reviewed version available at Sustainability 2018, 10, 2614; doi:10.3390/su10082614

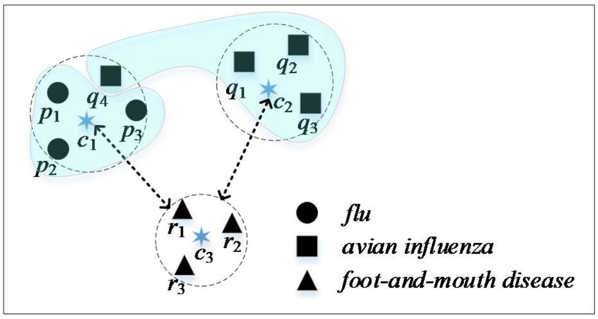

Figure 1. An example of clustering using location information

We present an example of disease analysis clustered using only location information. Figure 1 shows 10 cases of disease location divided into three clusters. Diseases are divided into three categories (i.e. *p*, *q* and *r*), and the measures to be taken in the area vary according to each disease. In Figure 1, the representative attribute of cluster c_1 is set to *p*, so we only deal with *p*. Therefore, when *q* occurs in the same area, it is difficult to cope with. In this example, three clusters are constructed using only numeric data (location information of disease occurrence). If the data from this example was used to inform a policy decision, it could result in a decision maker failing to implement the correct policy.

Data generated in the real world is often mixed with numeric data as well as categorical data. In order to apply the clustering technique to the real world, algorithms that can consider categorical data are required. A representative clustering algorithm that can use mixed data is the k-prototypes algorithm [15]. The basic k-prototypes algorithm has a large time complexity due to the processing of all data. Therefore, it is important to reduce execution time in order to process the k-prototypes algorithm on large data. However, only a few studies have been conducted to reduce the time complexity of the k-prototypes algorithm. Kim [16] proposed a pruning technique to reduce distance computation between an object and cluster centers using the concept of partial distance computation. However, this method does not have high pruning efficiency by comparing objects one by one with cluster center.

To improve performance, we propose an effective grid-based k-prototypes algorithm, GK-prototypes, for clustering spatial objects. The proposed method makes use of the grid-based indexing technique which improve pruning efficiency to compare distance between cluster centers and a cell instead of cluster centers and an object.

Spatial data can have geographic data as categorical attributes that indicate the characteristics of the object as well as the location of the object. Geographic data tend to have spatial dependence. Spatial dependence is the property of objects that are close to each other having increased similarities [17]. For example, soil types or network type are more likely to be similar at points one meter apart than at points one kilometer apart. Due to the nature of spatial dependence, the categorical data of spatial data is often skewed according to the position of the object. For improving performance of a grid-based k-prototypes algorithm, we take advantage of the spatial dependence to the bitmap indexing technique.

The contributions of this paper are summarized as follows.

- We proposed an effective grid-based k-prototypes, GK-prototypes, which improve the performance of a basic k-prototypes algorithm.
- We developed a pruning technique which utilizes the minimum and maximum distance on numeric attributes and the maximum distance on categorical attributes between a cell and a cluster center.
- We developed a pruning technique based on a bitmap index to improve the efficiency of the pruning in case that a categorical data is skewed.
- We conducted several experiments on synthetic datasets. Our algorithms can achieve better performance than the existing pruning technique in the k-prototypes algorithm.

The organization of the rest of this paper is as follows. In Section 2, the basic k-prototypes algorithm and the previous research on pruning in the k-prototypes algorithm are described. In Section 3, we first briefly describe some basic notations and definitions. After that, the proposed GK-prototypes algorithm is explained in Section 4. In Section 5, experimental results on synthetic data demonstrate the performance. Section 6 concludes the paper.

2. Related works

2.1. The k-prototypes algorithm

The k-prototypes algorithm is first proposed clustering algorithm to deal with mixed data types (numeric data and categorical data), which integrates k-means and k-modes algorithms [15]. Let a set of n objects be $O=\{o_1, o_2, \ldots, o_n\}$ where $o_i=(o_{i1}, o_{i2}, \ldots, o_{im})$ is consisted of m attributes. The purpose of clustering is to partition n objects into k disjoint clusters $C=\{C_1, C_2, \ldots, C_n\}$ according to the degree of similarity of objects. The distance is used as a measure to group objects with high similarity into the same cluster. The distance $d(o_i, C_j)$ between o_i and C_j is calculated as follows:

$$d(o_i, C_j) = d_r(o_i, C_j) + d_c(o_i, C_j) \quad (1)$$

where $d_r(o_i, C_j)$ is the distance between numeric attributes and $d_c(o_i, C_j)$ is the distance between categorical attributes.

$$d_r(o_i, C_j) = \sum_{k=1}^{p} |o_{ik} - c_{jk}|^2 \quad (2)$$

$$d_c(o_i, C_j) = \sum_{k=p+1}^{m} \delta(o_{ik}, c_{jk}) \quad (3)$$

$$\delta(o_{ik}, c_{jk}) = \begin{cases} 0, & when \quad o_{ik} = c_{jk} \\ 1, & when \quad o_{ik} \neq c_{jk} \end{cases} \quad (4)$$

In Equation (2), $d_r(o_i, C_j)$ is the squared Euclidean distance between an object and a cluster center on the numeric attributes. $d_c(o_i, C_j)$ is the dissimilar distance on the categorical attributes, where o_{ik} and c_{jk}, $1 \le k \le p$, are values of numeric attributes, o_{ik} and c_{jk}, $p+1 \le k \le m$ are values of categorical attributes. That is, p is the number of numeric attributes and m-p is the number of categorical attributes.

2.2. Existing pruning technique in the k-prototypes algorithm

The k-prototypes algorithm spends most of execution time computing the distance between an object and cluster centers. In order to improve the performance of the k-prototypes algorithm, Kim [16] proposed the concept of partial distance computation (PDC) which compares only partial attributes, not all attributes in measuring distance. The maximum distance that can be measured in one categorical attribute is 1. Thus the distance that can be measured from the categorical attributes is bound to the number of categorical attributes, m-p. Given an object o and the two cluster centers (c_1 and c_2), if the difference between $d_r(o, c_1)$ and $d_r(o, c_2)$ is more than m-p, we can know which clusters are closer to the object without the distance using the categorical attributes. However, PDC is still not efficient due to the fact that all objects are involved in the distance calculation and the characteristic (i.e. spatial dependence) of spatial data is not utilized in the clustering process.

2.3. Grid-based clustering algorithm

The grid-based techniques have the fastest processing time that depends on the number of the grid cells instead of the number of objects in the data set [18]. The basic grid-based algorithm is as follows. At first, a set of grid-cells is defined. In general, these grid-based clustering algorithm use a single uniform or multi-resolution grid cell to partition the entire datasets into cells. Each object is assigned to the appropriate grid cell and the density of each cell is computed. The cells, whose a

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 27 June 2018 doi:10.20944/preprints201806.0440.v1

Peer-reviewed version available at Sustainability 2018, 10, 2614; doi:10.3390/su10082614

degree of density is below a certain threshold, are eliminated. In addition to the density of cells, statistical information of objects in the cell is computed. After that, the clustering process is performed on the grid cells using each cell's statistical information, instead of the objects itself.

The representative grid-based clustering algorithms are STING [19] and CLIQUE [20]. STING is a grid-based multi resolution clustering algorithm in which the spatial area is divided into rectangular cells with a hierarchical structure. Each cell at a high level is divided into several smaller cells in the next lower level. For each cell in pre-selected layer, the relevancy of the cell is checked by computing the confidence interval. If the cell is relevant, we include the cell in a cluster. If the cell is irrelevant, it is removed from further consideration. We look for relevant cells at the next lower layer. This algorithm combines relevant cells into relevant regions and return the so obtained clusters. CLIQUE is a grid-based and density-based clustering algorithm to identify subspaces of a high dimensional data that allow better clustering quality than original data. CLIQUE partitions the n-dimensional data into non overlapping rectangular units. The units are obtained by partitioning every dimension into certain intervals of equal length and selectivity of a unit is defined as the total data points contained in it. A cluster in CLIQUE is a maximal set of connected dense units within a subspace. In a grid-based clustering study, the grid is used in order that clustering is performed on the grid cells, instead of objects itself. Chen et al. [21] proposes algorithm called GK-means, which integrates grid structure and spatial index with k-means algorithm. It focuses on choice the better initial centers to improve the clustering quality and to reduce the computational complexity of k-means.

Most existing grid-based clustering algorithms regard objects in same cell of grid as a data point to process large scale data. Thus, the final clustering results of these algorithms are not the same as a basic k-prototype cluster result, but all the cluster boundaries are either horizontal or vertical. In GK-means, the grid is used to select initial centers and remove noise data, but not used to reduce unnecessary distance calculation. To the best of our knowledge, such a grid-based pruning technique to improve the performance of the k-prototypes algorithm has not been previously demonstrated.

3. Preliminary

In this section, we present some basic notations and definitions before describing our algorithms. We summarize the notation used throughout this paper in Table 1.

Table 1. A summary of notations

Notation	Description
O	a set of data
o_i	i-th data in O
n	the number of objects
m	the number of attributes of an object
c_i	the i cluster center point
v_i	the value of grid partition interval
g_k	a cell of grid
$d(o_i, c_j)$	a distance between an object and an cluster center
$d_r(o_i, c_j)$	a distance between an object and an cluster center for only numeric attributes
$d_c(o_i, c_j)$	a distance between an object and an cluster center for only categorical attributes
$d_{min}(g_i, c_j)$	the minimum distance between a cell and a cluster center for only numeric attributes
$d_{max}(g_i, c_j)$	the maximum distance between a cell and a cluster center for only numeric attributes

Consider a set of n objects, $O=\{o_1, o_2, \ldots, o_n\}$. $o_i=(o_{i1}, o_{i2}, \ldots, o_{im})$ is an object represented by m attribute values. The m attributes consist of m_r (the number of numeric attributes) and m_c (the number of categorical attributes), $m=m_r+ m_c$. The distance between an object and a cluster center, $d(o, c)$, is calculated by Equation (1) in Section 2. We adopt the data indexing technique based on grid for pruning. First we define the cells that make up the grid.

Definition 1. A cell g in m_r-dimension grid is defined by a start point vector S and an end point vector T: $g = (S, T)$, where $S = [s_1, s_2, \ldots, s_{mr}]$ and $T = [t_1, t_2, \ldots, t_{mr}]$ and $s_i \le t_i$ for $1 \le i \le m_r$ and $s_i + v_i = t_i$. The v_i is interval distance between start and end position of a cell g on i-dimension.

Definition 2. The minimum distance between a cell g_i and a cluster center c_j for numeric attributes, denoted $d_{min}(g_i, c_j)$, is;

$$d_{min}(g_i, c_j) = \sqrt{\sum_{i=1}^{m_r} |o_i - r_i|^2},$$

where $r_i = \begin{cases} s_i & if\ o_i < s_i \\ t_i & if\ o_i > t_i \\ o_i & otherwise. \end{cases}$

We use the classic Euclidian distance to measure the distance. If a cluster center is inside the cell, the distance between them is zero. If a cluster center is outside the cell, we use the Euclidean distance between the cluster center and the nearest edge of the cell.

Definition 3. The maximum distance between a cell g_i and a cluster center c_j for numeric attributes, denoted $d_{max}(g_i, c_j)$, is;

$$d_{max}(g_i, c_j) = \sqrt{\sum_{i=1}^{m_r} |p_i - r_i|^2},$$

where $r_i = \begin{cases} t_i, & p_i \le \frac{s_i + t_i}{2} \\ s_i, & otherwise. \end{cases}$

To distinguish between the two distances d_{min} and d_{max}, an example is illustrated in Figure 2, showing a cluster center (c_1), two cells (g_1 and g_2) and the corresponding distances.

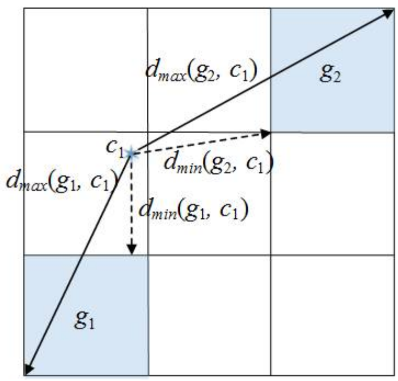

Figure 2. An example of distances d_{min} and d_{max}.

In a grid-based approach, $d_{min}(g,c)$ and $d_{max}(g,c)$ for a cell g and cluster centers c, are measured firstly before measuring the distance between an object and cluster centers. We can use d_{min} and d_{max} to improve performance of k-prototypes algorithm. Figure 3 shows an example of pruning using d_{min} and d_{max}. The object consists of 4 attributes (2 numeric and 2 categorical attributes).

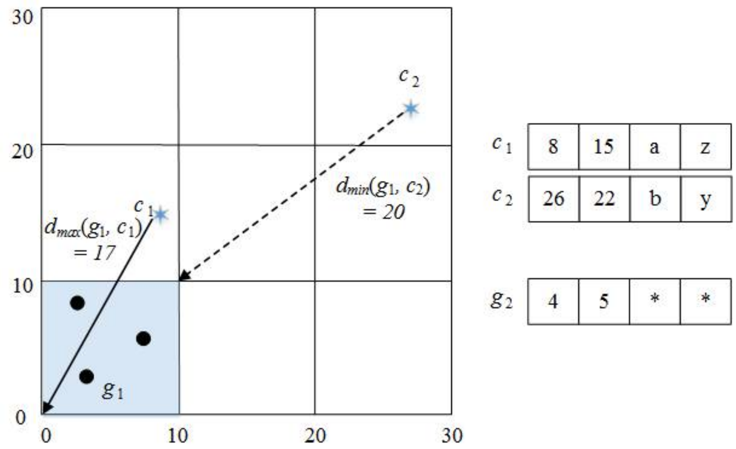

Figure 3. An example of pruning method using minimum distance and maximum distance

In Figure 3, the maximum distance between g_1 and c_1, $d_{max}(g_1, c_1) = 17(\sqrt{(8-0)^2+(15-0)^2})$. The minimum distance between g_1 and c_2, $d_{min}(g_1, c_2) = 20$ ($\sqrt{(26-10)^2+(22-10)^2}$). The $d_{max}(g_1, c_1)$ is three less than $d_{min}(g_1, c_2)$. Therefore, all objects in g_1 are closer to c_1 than c_2, if we are considering only numeric attributes. To find the closest cluster center from an object, we have to measure the distance by categorical attributes. If the difference between d_{min} and d_{max} is more than m_c (maximum distance by categorical attributes), however, the cluster closest to the object can be determined without the distance by categorical attributes. In Figure 3, the categorical data of c_1 is (a, z), and the categorical data of c_2 is (b, y). Assume that there are no objects in g_1 with 'a' in the first categorical attribute and 'z' in second categorical attribute. Since $d_c(o, c_1)$ of all objects in g_1 is 2, maximum distance of all objects in g_1 is $d_{max}(g_1, c_1) + 2$. The maximum distance between c_1 and objects in g_1 is not less than the minimum distance between c_2 and objects in g_1. We can know that all objects in g_1 are closer to c_1 than c_2.

Lemma 1. For any cluster centers c_i, c_j and any cell g_x, if $d_{min}(g_x, c_j) - d_{max}(g_x, c_i) > m_c$ then, $\forall o \in g_x$, $d(o, c_i) < d(o, c_j)$.

Proof. By assumption, $d_{min}(g_x, c_j) > m_c + d_{max}(g_x, c_i)$. By Definition 2 and 3, $d_{min}(g_x, c_i) \le d(c_i, o) \le d_{max}(g_x, c_i) + m_c$, and $d_{min}(g_x, c_j) \le d(c_j, o) \le d_{max}(g_x, c_j) + m_c$. $d(c_i, o) \le d_{max}(g_x, c_i) + m < d_{min}(g_x, c_j) \le d(c_j, o)$. $\therefore \forall o \in g_x$, $d(c_i, o) < d(c_j, o)$ □.

Lemma 1 is the basis for our proposed pruning techniques. In the process of clustering, we first exploit Lemma 1 to remove cluster centers to be compared to objects.

4. GK-prototypes algorithm

In this section, we present two pruning techniques that are based on grid for improving the performance of the k-prototypes algorithm. The first pruning technique is KCP (K-prototypes algorithm with Cell Pruning) which utilizes d_{min}, d_{max} and the maximum distance on categorical attributes. The second pruning technique is KBP (K-prototypes algorithm with Bitmap Pruning) which utilizes bitmap indexes to reduce unnecessary distance calculation on categorical attributes.

4.1. Cell pruning technique

The computational cost of the k-prototypes algorithm is most often encountered in the step of measuring distance between objects and cluster centers. To improve the performance, each object is indexed into a grid by numeric data in data preparation step. We set up grid cells storing two types of information. a) The first is a start point vector *S* and an end point vector *T*, which is the range of the numeric value of the objects to be included in the cell (see Definition 1). Based on this cell information, the minimum and maximum distances between each cluster centers and a cell are measured. b) The second is bitmap indexes which is explained in Subsection 4.2.

Algorithm 1 The k-prototypes algorithm with cell pruning (KCP)
Input: k: the number of cluster, G: the grid in which all objects are stored per cell
Output: *k* cluster centers

```
1: C[ ]← Ø    // k cluster centers
2: Randomly choosing k object, and assigning it to C.
3: while IsConverged() do
4:     for each cell g in G
5:         dmin[ ], dmax[ ] ← Calc(g, C)
6:         dminmax ← min(dmax[ ])
7:         candidate ← Ø
8:         for j ←1 to k do
9:             if (dminmax + mc > dmin[j] )   // Lemma 1
10:                candidate ← candidate ∪ j
11:        end for
12:        min_distance ← ∞
13:        min_cluster ← null
14:        for each object o in g
15:            for each center c in candidate
16:                if min_distance > d(o, c)
17:                    min_cluster ← index of c
18:            end for
19:            Assign(o, min_cluster)
20:            UpdateCenter(C[c])
21:        end for
22: end while
23: return C[k]
```

The details of the cell pruning are as follows. First, we initialize an array $C[j]$ to store the position of k cluster centers, $1 \leq j \leq k$. Various initial cluster centers selection methods have been studied to improve the accuracy of clustering results in the k-prototypes algorithm [22]. Since we aim at increasing the clustering speed improvement, however, we adopt a simple method to select k objects randomly from input data and use them as initial cluster centers.

In general, the result of clustering algorithm is evaluated after a single clustering process has been performed. Based on these evaluation result, it is determined whether the same clustering process have to be repeated or terminated. The iteration is terminated if the sum of the difference between the current cluster center and the previous cluster center is less than a predetermined value (ε) as an input parameter by users. In Algorithm 1, we determine the termination condition through the *IsConverged*() function of the while statement.

In iteration step of clustering (line 4), the distances between objects and k cluster centers are measured by cells. The *Calc*(g, C) function returns the minimum and maximum distances between a cell g and k cluster centers (C) for each cluster center (line 5). The smallest distance among the maximum distance is stored in the *dminmax* (line 6). The candidate stores the index number of the cluster center that need to be measured from the cell. Through Lemma 1, if $dminmax + m_c$ is greater than $dmin[j]$, the j cluster center is included in the distance calculation, otherwise it is excluded (lines 8-11). After Lemma 1 is applied, only the cluster centers to be measured the distance from objects in the cell are finally left in the candidate. All objects in the cell are measured from the cluster centers in the candidate, $d(o, c)$, (line 16). An object is assigned to the cluster where $d(o, c)$ is computed as the smallest value using *Assign*(o, *min_cluster*) function. The center of cluster to which a new object is added is updated by remeasuring its cluster center using *UpdateCenter*($C[c]$) function. This clustering process is repeated until the end condition *IsConverged*() is return a false.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 27 June 2018 doi:10.20944/preprints201806.0440.v1

Peer-reviewed version available at *Sustainability* **2018**, *10*, 2614; doi:10.3390/su10082614

4.2. Bitmap pruning technique

Spatial data tend to have similar categorical values in neighboring objects, categorical data is often skewed. If we can utilize the characteristics of spatial data, the performance of the k-prototypes algorithm can be further improved. In this subsection, we introduce the KBP that can improve the efficiency of pruning when categorical data is skewed.

The KBP stores categorical data of all objects in a cell in a bitmap index. Figure 4 shows an example of storing categorical data as a bitmap index. Figure 4(a) is an example of spatial data. The x and y attributes indicate location information of objects as numeric data, and z and w attributes indicate features of objects as categorical data. For five objects in the same cell, g={o_1, o_2, o_3, o_4, o_5}, Figure 4(b) shows the bitmap index structure where a row presents a categorical attribute and a column presents a categorical data of objects in same cell. A bitmap index consists of one vector of bits per attribute value, where the size of each bitmap is equal to the number of categorical data in the raw data. The bitmaps are encoded such that the *i*-th bit is set to 1 if the raw data has a value corresponding to the *i*-th column in the bitmap index, otherwise it is set to 0. For example, the value 1 in z row and c column from bitmap index means that the value c exists in z attribute of raw data. When raw data is converted to a bitmap index, object id information is removed. We can quickly check for the existence of the specified value in raw data using the bitmap index.

oid	*x*	*y*	*z*	*w*
o_1	*14*	*6*	*A*	*D*
o_2	*12*	*3*	*F*	*B*
o_3	*30*	*4*	*C*	*D*
o_4	*31*	*9*	*F*	*C*
o_5	*59*	*9*	*F*	*E*

(a) Example mixed data

	A	*B*	*C*	*D*	*E*	*F*
z	*1*	*0*	*1*	*0*	*0*	*1*
w	*0*	*1*	*1*	*1*	*1*	*0*

(b) Bitmap indexing sturucture

Figure 4. Bitmap indexing structure

A maximum of categorical distance is determined by the number of categorical attributes (m_c). If the difference of two numeric distance between one object and two cluster centers, $|dr(o, c_i) - dr(o, c_j)|$, is more than m_c, we can know the cluster center closer to the object without categorical distance. However, since the numeric distance is not known in advance, we cannot determine the cluster center closer to the object by only categorical distance. Thus, the proposed KBP is utilized in reducing categorical distance calculations in the KCP. Algorithm 2 describes the proposed KBP. We explain only the extended parts from the KCP.

Algorithm 2 The k-prototypes algorithm with bitmap pruning (KBP)
Input: k: the number of cluster, G: the grid in which all objects are stored per cell
Output: *k* cluster centers

$C[k] \leftarrow \varnothing$ // *k* cluster center
Randomly choosing *k* object, and assigning it to *C*.
while *IsConverged*() **do**
    for each cell *g* **in** *G*
        dmin[], *dmax*[] ← *Calc*(*g*, *C*)
        dminmax ← *min*(*dmax*[])
        arrayContain[] ← *IsContain*(*g*, *C*)
        candidate ← ∅
        for *j* ←1 **to** *k* **do**
            if ($dminmax + m_c > dmin[j]$) // Lemma 1

```
11:             candidate ← candidate ∪ j
12:     end for
13:     min_distance ← ∞
14:     min_cluster ← null
15:     distance ← null
16:     for each object o in g
17:         for each center c in candidate
18:             if (arrayContain [c] == 0)
19:                 distance = d_r(o, c) + m_c
20:             else
21:                 distance = d_r(o, c) + d_c(o, c)
22:             if min_distance > distance
23:                 min_cluster ← index of cluster center
24:         end for
25:         Assign(o, min_cluster)
26:         UpdateCenter(C)
27:     end for
28: end while
29: return C[k]
```

To improve the efficiency of pruning on categorical data, the KBP is implemented by adding two step to the KCP. The first step is to find out whether the categorical attributes value of each cluster centers exists in the bitmap index of the cell using the *IsContain* function (line 7). The *IsContain* function compares the categorical data of each cluster center with the bitmap index and returns 1 if there is more than one of the same data in the corresponding attribute. Otherwise, it returns 0. In Figure 4 (b), assume that we have a cluster center, c_i = (2, 5, D, A). The bitmap index does not have D in the z attribute and A in the w attribute. In this case, we can know that there are no objects in the cell that have D in the z attribute and A in the w attribute. Therefore, the maximum categorical distance between all objects belonging to a cell and cluster center ci is 0. Assume that we have another the cluster center, c_j = (2, 5, A, B). The bitmap index has A in the z attribute and B in the w attribute. In this case, we can know that there are some objects in the cell that have A in the z attribute or b in the w attribute. If one or more objects with the same categorical value are found in the corresponding attribute, the categorical distance calculation has to be performed for all objects in the cell in order to know the correct categorical distance. Finally, *arrayContain*[*i*] stores the result of the comparison between the *i*-th cluster center and the cell. In lines 8-12, the cluster centers that need to measure distance with each cell *g* are stored in the candidate like as the KCP. The second extended part (lines 18-23) is used to determine whether to measure the categorical distance using *arrayContain*. If *arrayContain*[*i*] has 0, the m_c is directly used as the result of the categorical distance without measuring the categorical distance (line 19).

5. Experiments

In this section, we evaluate the performance of the proposed GK-prototypes algorithms. For performance evaluation, we compare partial distance computation pruning (PDC) [16] and our two algorithms (KCP and KBP). We examined the performance of our algorithms as the number of objects, the number of clusters, the number of categorical attributes and the number of division in each dimension increased. In Table 2, the parameters used for the experiments are summarized. The rightmost column of the table means the baseline values of the various parameters. For each set of parameters, we perform 10 sets of experiments and the average values are reported. Even under the same parameter values, the number of clustering iterations will vary if the input data is different.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 27 June 2018 doi:10.20944/preprints201806.0440.v1

Peer-reviewed version available at Sustainability 2018, 10, 2614; doi:10.3390/su10082614

Therefore, we measure the performance of each algorithm with the average execution time that it takes for clustering to repeat once.

Table 2. Parameters for the experiments

Parameter	Description	Baseline Value
n	no. of objects	1,000K
k	no. of clusters	10
m_r	no. of numeric attributes	2
m_c	no. of categorical attributes	5
s	no. of division in each dimensions	10

The experiments are carried out on a PC with Intel(R) Core(TM) i7 3.5 GHz, 32GB RAM. All the algorithms are implemented in Java.

5.1. Data sets

We generate many synthetic datasets with numeric attributes and categorical attributes. For numeric attributes in each dataset, two numeric attributes are generated in the 2D space [0, 100]×[0, 100] to indicate an object's location. Each object is assigned into $s \times s$ cells in a grid by these numeric data. The numeric data is generated according to uniform distributions in which each numeric data is selected in [0, 100] randomly or Gaussian distributions with mean = 50 and standard deviation = 10. The categorical data is generated according to uniform distributions or skewed distributions. For uniform distributions, we select an alphabet from A to Z randomly. For skewed distributions, we generate categorical data in such a way that objects in same cell have similar alphabet based on its numeric data.

5.2. Effects of the number of objects

To illustrate scalability, we vary the number of objects from 100K to 1,000K. Other parameters are given their baseline values (Table 2). Figures 5, 6 and 7 show the effect of the number of objects. Three graphs are shown in a linear scale. For each algorithm, the runtime per iteration is approximately proportional to the number of objects.

In Figure 5, KBP and KCP outperforms PDC. However, there is little difference in the performance between KBP and KCP. This is because if the categorical data is uniform distribution, most of the categorical data exist in the bitmap index. In this case, KBP is the same performance as KCP.

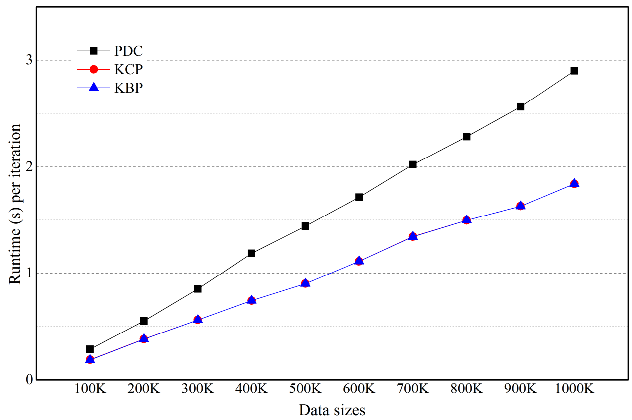

Figure 5. Effect of the number of objects (numeric data and categorical data are on uniform distribution)

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 27 June 2018 doi:10.20944/preprints201806.0440.v1

Peer-reviewed version available at Sustainability 2018, 10, 2614; doi:10.3390/su10082614

In Figure 6, KBP outperforms KCP and PDC. For example, KBP runs up to 1.1 and 1.75 times faster than KCP and PDC, respectively (n=1,000K). If categorical data is on a skewed distribution, KBP is effective for improving performance. As the size of the data increases, the difference in execution time increases. This is because as the data size increases, the amount of distance calculation increases, while at the same time the number of objects included in the cluster being pruned increases. In Figure 7, KCP outperforms PDC. Even if numeric data is on a Gaussian distribution, cell pruning is effective for improving performance.

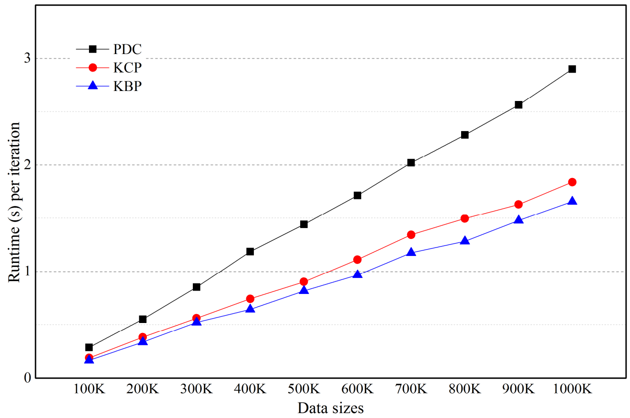

Figure 6. Effect of the number of objects (numeric data is on uniform distribution and categorical data is on skewed distribution)

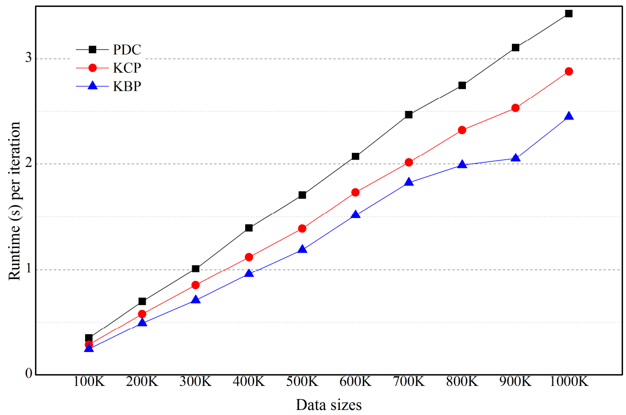

Figure 7. Effect of the number of objects (numeric data is on Gaussian distribution and categorical data is on skewed distribution)

5.3. Effects of the number of clusters

To confirm the effects of the number of clusters, we vary the number of clusters, k, from 5 and 20. Other parameters are kept at their baseline values (Table 2). Figures 8, 9 and 10 show the effects of the number of clusters. Three graphs are also shown in a linear scale. For each algorithm, the runtime is approximately proportional to the number of cluster.

In Figure 8, KBP and KCP also outperform PDC. However, there is also little difference in the performance between KBP and KCP. This is because if the categorical data is uniform distribution, most of the categorical data exist in the bitmap index like Figure 5.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 27 June 2018 doi:10.20944/preprints201806.0440.v1

Peer-reviewed version available at Sustainability 2018, 10, 2614; doi:10.3390/su10082614

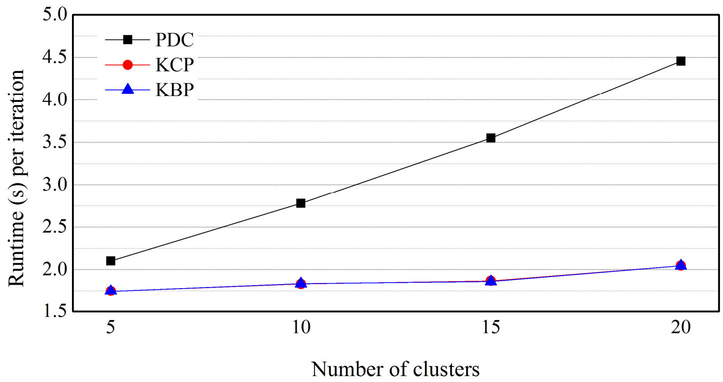

Figure 8. Effect of the number of clusters (numeric data and categorical data are on uniform distribution)

In Figure 9, KBP outperforms KCP and PDC. For example, KBP runs up to 1.13 and 1.71 times faster than KCP and PDC, respectively (k = 10). This result indicates that KBP is effective for improving performance even if categorical data is on a skewed distribution. As the number of cluster increases, the difference in execution time increases. This is also because as the data size increases, the amount of distance calculation increases, while at the same time the number of objects included in the cluster being pruned increases like Figure 6. In Figure 10, KCP also outperforms PDC. Even if numeric data is on a Gaussian distribution, KCP is also effective for improving performance.

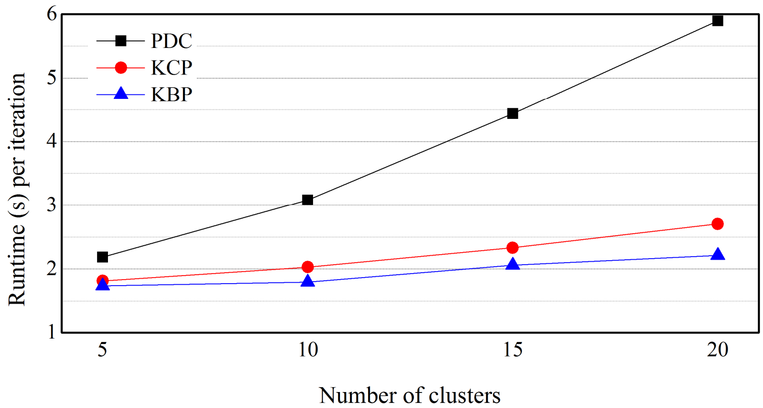

Figure 9. Effect of the number of clusters (numeric data and categorical data are on uniform distribution)

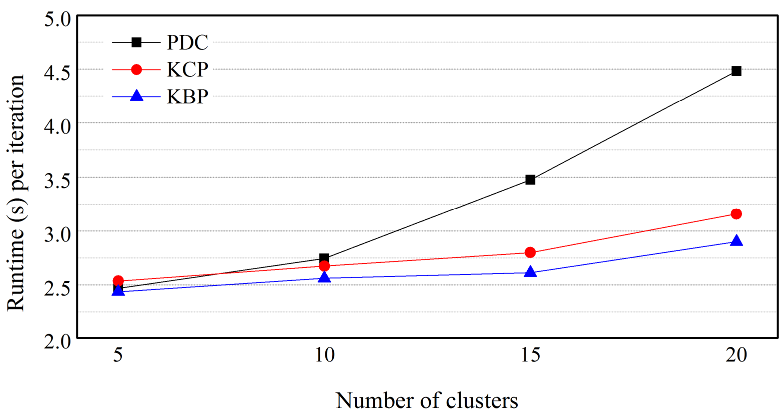

Figure 10. Effect of the number of clusters (numeric data is on Gaussian distribution and categorical data is on skewed distribution)

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 27 June 2018 doi:10.20944/preprints201806.0440.v1

Peer-reviewed version available at Sustainability 2018, 10, 2614; doi:10.3390/su10082614

5.4. Effects of the number of categorical attributes

To confirm the effects of the number of categorical attributes, we vary the number of categorical attributes from 5 to 20. Other parameters are given their baseline values. Figure 11 shows the effect of the number of categorical attributes. The graph is also shown in a linear scale. For each algorithm, the runtime per iteration is approximately proportional to the number of categorical attributes. KBP outperforms KCP and PDC. For example, KBP runs up to 1.13 and 1.71 times faster than KCP and PDC, respectively (m_c=5). Even if the number of the categorical attributes increases, the difference between the execution time of KCP and PDC is kept almost constant. The reason is that KCP is based on numeric attributes and is not affected by the number of categorical attributes.

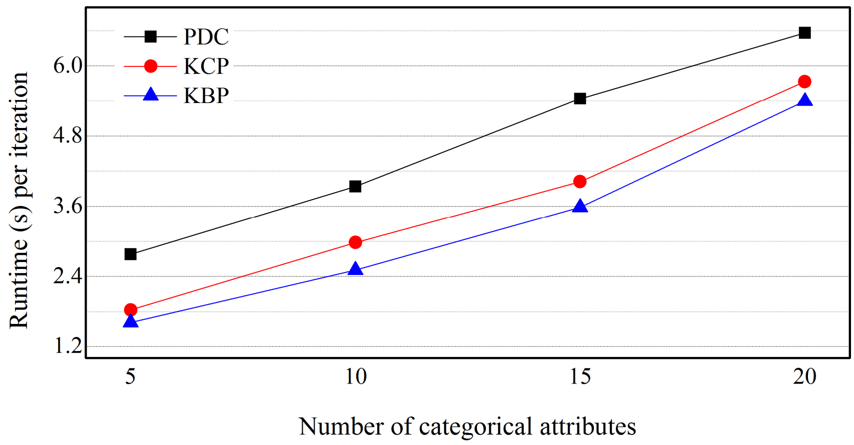

Figure 11. Effect of the number of categorical attributes on a skewed distribution

5.5. Effects of the size of cells

To confirm the effects of the size of cells, we vary the number of cells from 5 to 20. Other parameters are given their baseline values (Table 2). Figure 12 shows the effect of the number of divisions in each dimension. KBP and KCP outperform PDC. In Fig. 13, the horizontal axis is the number of divisions of each dimension. As the number of divisions increases, the size of the cell decreases. As the cell size gets smaller, the distance between the cell and cluster centers can be measured more finely. There is no significant difference in execution time according to the size of cell by each algorithm. This is because the distance calculation between the cell and the cluster centers is increased in proportion to the number of cells, and the bitmap index stored by the cell is also increased.

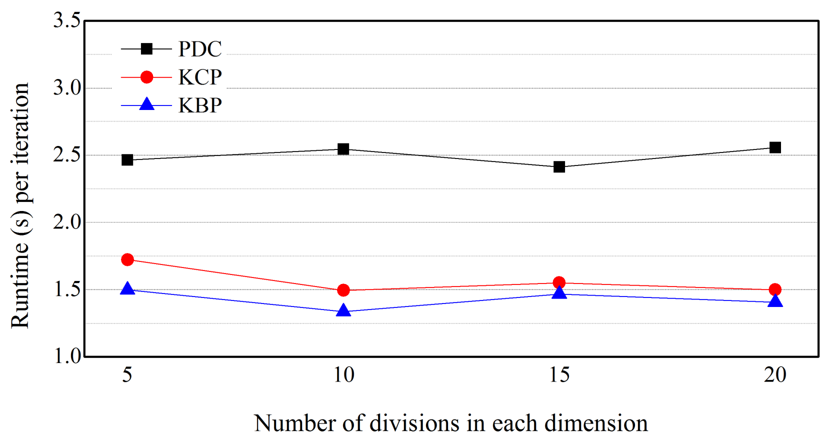

Figure 12. Effect of the number of cells on a skewed distribution

6. Conclusions

In this paper we have propose an efficient grid-based k-prototypes algorithm, GK-prototypes, that improves performance for clustering spatial objects. We develop two pruning techniques, KCP and KBP. KCP which uses both maximum and minimum distance between cluster centers and a cell improves the performance than PDC. KBP is an extension of cell pruning for improving the efficiency of pruning in case that a categorical data is skewed. Our experimental results demonstrate that KCP and KBP outperforms PDC, and KBP outperforms KCP except for uniform distributions of categorical data. These results lead us to conclude that our grid-based k-prototypes algorithm can achieve better performance than the existing k-prototypes algorithm.

As data has grown exponentially and more complex recently, the traditional clustering algorithms have a great challenge to deal with these data. In future works, we may consider optimized pruning techniques of the k-prototypes algorithm in parallel processing environment.

Author Contributions: Conceptualization, Hong-Jun Jang and Byoungwook Kim; Methodology, Hong-Jun Jang; Software, Byoungwook Kim; Writing-Original Draft Preparation, Hong-Jun Jang and Byoungwook Kim; Writing-Review & Editing, Jongwan Kim; Supervision, Soon-Young Jung.

Acknowledgments: This research was supported by Basic Science Research Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Education(No. NRF-2017R1D1A1B03034067) and by the National Research Foundation of Korea(NRF) grant funded by the Korea government(MSIT) (No. NRF-2016R1A2B1014013)

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Sander, J.; Ester, M.; Kriegel, H.P.; Xu, X. Density-Based Clustering in Spatial Databases: The Algorithm GDBSCAN and Its Applications. *Data Mining and Knowledge Discovery* **1998,** *2*, 169–194.
2. Koperski, K.; Han, J.; Stefanovic, N. An Efficient Two-Step Method for Classification of Spatial Data. In Proceedings of the International Symposium on Spatial Data Handling (SDH'98), Vancouver, Canada, 1998, pp. 45–54.
3. Koperski, K.; Han. J. (1995). Discovery of Spatial Association Rules in Geographic Information Databases. In Proceedings of the 4th International Symposium on Advances in Spatial Databases (SSD'95), 1995, pp. 47–66.
4. Ester, M.; Frommelt, A.; Kriegel, H.P.; Sander, J. Algorithms for Characterization and Trend Detection in Spatial Databases. In Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining (KDD'98), 1998, pp. 44–50.
5. Deren, L.; Shuliang, W.; Wenzhong, S.; Xinzhou, W. On Spatial Data Mining and Knowledge Discovery. *Geomatics and Information Science of Wuhan Univers* **2001**, *26*, 491–499.
6. Boldt, M.; Borg, A. A statistical method for detecting significant temporal hotspots using LISA statistics. In Proceedings of the Intelligence and Security Informatics Conference (EISIC), 2017 European, 2017.
7. Chainey, S.; Reid, S.; Stuart, N. When is a Hotspot a Hotspot? A Procedure for Creating Statistically Robust Hotspot Maps of Crime. In Innovations in GIS 9: Socio-economic applications of geographic information science, Kidner, D; Higgs, G; White, S, Eds.; Taylor & Francis: London, UK, 2002; pp. 21–36.
8. Murray, A.; McGuffog, I.; Western, J.; Mullins, P. Exploratory spatial data analysis techniques for examining urban crime. *The British Journal of Criminology* **2001,** *41*, pp. 309–329.
9. Chainey, S.; Tompson, L.; Uhlig, S. The Utility of Hotspot Mapping for Predicting Spatial Patterns of Crime. *Security Journal* **2008,** *21*, pp. 4–28.
10. Di Martino, F.; Sessa, S. The extended fuzzy C-means algorithm for hotspots in spatio-temporal GIS. *Expert Systems with Applications* **2011**, *38*, pp. 11829–11836.
11. Di Martino, F.; Sessa, S.; Barillari, U.E.S.; Barillari, M.R. Spatio-temporal hotspots and application on a disease analysis case via GIS. *Soft Computing* **2014,** *18*, pp. 2377–2384.
12. Mullner, R.M.; Chung, K.; Croke, K. G.; Mensah, E. K. Geographic information systems in public health and medicine. *Journal of Medical Systems* **2004,** *28*, pp. 215–221.
13. Polat, K. Application of attribute weighting method based on clustering centers to discrimination of linearly non-separable medical datasets. *Journal of Medical Systems* **2012,** *36*, pp. 2657–2673.

14. Wei, C.K.; Su, S.; Yang, M.C. Application of data mining on the development of a disease distribution map of screened community residents of taipei county in Taiwan. *Journal of Medical Systems* **2012,** *36*, pp. 2021–2027.
15. Huang, Z. Clustering large data sets with mixed numeric and categorical values. In Proceedings of the First Pacific Asia Knowledge Discovery and Data Mining Conference, 1997, pp. 21–34.
16. Kim, B. A Fast K-prototypes Algorithm Using Partial Distance Computation. *Symmetry* **2017,** *9*, doi:10.3390/sym9040058
17. Goodchild, M. Geographical information science. *International Journal of Geographic Information Systems* **1992,** *6*, pp. 31–45.
18. Xiaoyun, C.; Yi, C.; Xiaoli, Q.; Min, Y.; Yanshan, H. PGMCLU: a novel parallel grid-based clustering algorithm for multi-density datasets. In: 1st IEEE symposium on web society, 2009 (SWS'09), Lanzhou, 2009, pp 166–171.
19. Wang, W.; Yang, J.; Muntz, R. R. STING: A Statistical Information Grid Approach to Spatial Data Mining. In the 23rd International Conference on Very Large Data Bases (VLDB'97), 1997, pp. 186–195.
20. Agrawal, R.; Gehrke, J.; Gunopulos, D.; Raghavan, P. Automatic Subspace Clustering of High Dimensional Data for Data Mining Applications. In Proceedings of the ACM SIGMOD International Conference on Management of Data, ACM Press, 1998, pp. 94–105.
21. Chen, X., Su, Y., Chen, Y., & Liu, G. GK-means: An Efficient K -means Clustering Algorithm Based On Grid. In Computer Network and Multimedia Technology (CNMT 2009) International Symposium, 2009.
22. Ji, J.; Pang, W.; Zheng, Y.; Wang, Z.; Ma, Z.; Zhang, L. A Novel Cluster Center Initialization Method for the k-Prototypes Algorithms using Centrality and Distance. *Applied Mathematics & Information Sciences* **2015,** *9*, pp. 2933–2942.
23. Zavadskas, E.K.; Antucheviciene, J.; Vilutiene, T.; Adeli, H. Sustainable Decision Making in Civil Engineering, Construction and Building Technology. *Sustainability* **2018**, 10, 14.
24. Hersh, M.A. Sustainable Decision Making: The Role of Decision Support systems. *IEEE Transactions on Systems, Man, and Cybernetics-Part C: Applications and Reviews* **1999**, 29, 3, pp. 395-408.
25. Morik, K.; Bhaduri, K.; Kargupta. H. Introduction to data mining for sustainability. *Data Mining and Knowledge Discovery* **2012**, 24, 2, pp. 311-324.
26. Aissi, S.; Gouider, M.S.; Sboui, T.; Said, L.B. A spatial data warehouse recommendation approach:conceptual framework and experimental evaluation. *Human-centric Computing and Information Sciences* **2015**, 5, 30.
27. Kim, J.-J. Spatio-temporal Sensor Data Processing Techniques. *Journal of Information Processing Systems* **2017**, 13, 5, pp. 1259-1276.