Int. J. Advance Soft Compu. Appl, Vol. 8, No. 3, December 2016
ISSN 2074-8523

Cell-DROS: A Fast Outlier Detection Method for Big Datasets

Duong Van Hieu[1] and Phayung Meesad[1]

[1]Faculty of Information Technology,
King Mongkut's University of Technology North Bangkok, Thailand
e-mail: dvhieu@gmail.com, duongvanhieu@tgu.edu.vn
pym@kmutnb.ac.th

Abstract

Outlier detection is one of the obstacles of big dataset analysis because of its time consumption issues. This paper proposes a fast outlier detection method for big datasets, which is a combination of cell-based algorithms and a ranking-based algorithm with various depths. A cell-based algorithm is proposed to transform a very large dataset to a fairly small set of weighted cells based on predefined lower and upper bounds. A ranking-based algorithm with various depths is modified and applied on weighted cells to calculate outlier scores and sort cells based on their outlier scores. Finally, an outlier obtaining algorithm is proposed to identify ids of outliers from the ranked cells and eliminate outliers from the given datasets. Experiment results show that the proposed method can produce the same results when compared to the previous rank-difference outlier detection algorithm but it can reduce up to 99% of executing time.

Keywords: *Cell-based outlier detection, Rank-based outlier detection, Outlier detection, Outlier detection in big datasets.*

1 Introduction

Outlier analysis is an important process in the data science field because of its applicability to various sorts of different problems including military surveillance, fault detection in safety systems, security intrusion detection, and credit card fraud detection, etc. [1]. An outlier is defined as an observation which deviates highly from the remaining observations [2]. The outlier objects may be generated by different sources or mechanisms compared to the remaining objects in the same datasets. Outlier objects or outliers are sometimes called abnormal, deviant, or discordant objects [3].

Outlier detection algorithms normally find models of normal objects from provided datasets. Hence, outliers are objects which do not fit those normal models. Results of outlier detection algorithms are normally in one of two forms. The former is the binary labelling results; each object is assigned a label 'normal' or 'outlier'. On the other hand, the later assigns a real value as outlier score to each object and sorts objects in a descending order based on the outlier scores. The top k sorted objects are considered as k outliers [4].

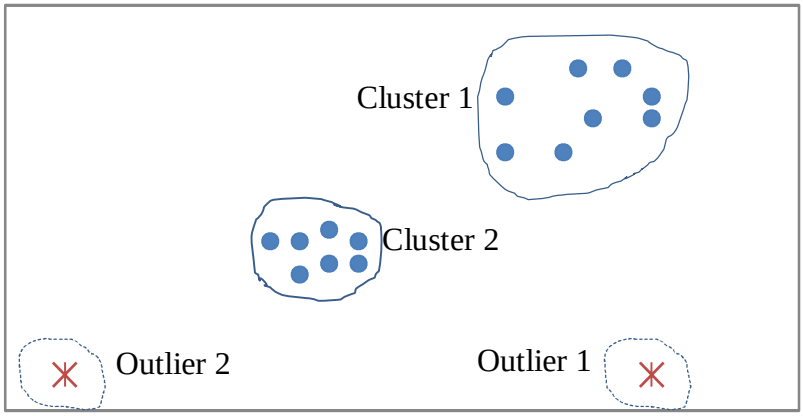


Fig. 1: An example of a dataset having 2 clusters and 2 outliers.

The well-known outlier detection algorithms including density-based local outlier factor (LOF) [5], connectivity-based outlier factor (COF) [6], influential measure of outliers by symmetric neighborhood relationships (INFLO) [7], rank-based detection algorithms (RBDA) [8], and ranking combined with clustering algorithms (ODMR, ODMRD) [9] have limitations in terms of using density. The LOF and COF algorithms which uses k nearest neighbors may produce poor results when outliers are located between low and high density clusters, the INFLO algorithm may produce incorrect outliers because it uses an assumption that all neighbors of an object have the same density, results of the RBDA, ODMR, ODMRD algorithms could be negatively affected by local irregularities of datasets.

Recently, to overcome the limitations of the aforementioned well-known outlier detection algorithms, a precise outlier detection method based on multi-time sampling [10] and a ranking-based outlier detection algorithm with various depths (RDOS) [11] have been proposed. It was reported that the newly proposed algorithms can produce very high precise results compared to the previous algorithms. However, these new algorithms cannot be applied to datasets having large numbers of objects due to very long execution time.

High precise outlier detection algorithms are usually extensively computational. When datasets have a very large number of objects, detection of outlier objects in those datasets is a challenging problem [12]. This paper proposes a fast outlier

detection method which is a hybrid of cell-based algorithms and the ranking-based outlier detection algorithm with various depths. This proposed algorithm (abbreviated to Cell-RDOS) should be an appropriate outlier detection method for datasets with a very large number of objects.

Contributions of the proposed Cell-RDOS method are very significant to outlier analysis in the big data era. Firstly, a cell-based method is proposed to transform a very large dataset into a fairly small set of weighted cells based on the predefined lower and upper magnitudes. Secondly, a revised version of the ranking-based outlier detection algorithm with various depths will be applied to the sets of weighted cells to calculate outlier scores of cells and rank cells based on the outlier scores. Finally, an outlier obtaining algorithm is proposed to identify outlier objects from the ranked cells and remove outliers from the provided datasets. The proposed Cell-RDOS method can produce the same results as the RDOS algorithm but it can reduce up to 99% execution time of the RDOS algorithm.

The rest of this paper is organized into four sections. Section 2 is related literatures. Section 3 explains the proposed Cell-DROS method. Section 4 covers experiments and results. Last but not least, conclusions and discussions are covered in Section 5.

2 Related Work

This paper focuses on the second form of outlier detection results. The outlier detection algorithms assign real values as outlier scores to objects and rank them in a descending order based on their outlier scores. The top k ranked objects are considered as k outliers. For instance, the object 122 in Table 1 should be considered as an outlier.

Table 1: An example of sorted objects associated with outlier scores.

Object IDs	Outlier Scores
122	1,429.75
18	211.79
207	120.34
177	96.93
192	82.82
91	70.01
…	…

The first algorithm belonging to this ranked-based strategy is to use local outlier factor (LOF) [5]. This density-based outlier ranking algorithm initially computes abnormal degrees of all objects based on the definitions of local reachability density and k-distance neighbors. Then, objects are ordered based on abnormal magnitudes. The top k objects with the highest abnormal magnitudes are

considered as top k outliers. Let $\mathbf{p}$, $\mathbf{o}$ be objects, k be a positive integral number, $\mathbf{X}$ be a set of objects, and *k-distance*($\mathbf{p}$) or $d(\mathbf{p}, \mathbf{o})$ is a distance between $\mathbf{p}$ and $\mathbf{o} \in \mathbf{X}$ such that:

1. Having at least k objects $\mathbf{o'} \in \mathbf{X} \setminus \{\mathbf{p}\}$ satisfy $d(\mathbf{p}, \mathbf{o'}) \leq d(\mathbf{p}, \mathbf{o})$,
2. And, having at most k-1 objects $\mathbf{o'} \in \mathbf{X} \setminus \{\mathbf{p}\}$ satisfy $d(\mathbf{p}, \mathbf{o'}) < d(\mathbf{p}, \mathbf{o})$.

k-distance neighbors of an object $\mathbf{p}$ are objects whose distances from $\mathbf{p}$ are not greater than *k-distance*($\mathbf{p}$) and denoted by (1).

$$N_{k\text{-distance}(\mathbf{p})} = \{\mathbf{q} \in \mathbf{X} \setminus \{\mathbf{p}\} | d(\mathbf{p}, \mathbf{q}) \leq k\text{-distance}(\mathbf{p})\} \quad (1)$$

Reachability of an object $\mathbf{p}$ with respect to an object $\mathbf{o}$ is defined by (2), local reachability density of an object $\mathbf{p}$ is defined by (3), and local outlier factor of an object $\mathbf{p}$ is defined by (4).

$$reach\text{-}dist_k(\mathbf{p}, \mathbf{o}) = \max\{k\text{-}distance(\mathbf{o}), d(\mathbf{p}, \mathbf{o})\} \quad (2)$$

$$lrd_k(\mathbf{p}) = \frac{1}{\frac{\sum_{\mathbf{o} \in N_k(\mathbf{p})} reach-dist_k(\mathbf{p},\mathbf{o})}{|N_k(\mathbf{p})|}} \quad (3)$$

$$LOF_k(\mathbf{p}) = \frac{\sum_{\mathbf{o} \in N_k(\mathbf{p})} \frac{lrd_k(\mathbf{o})}{lrd_k(\mathbf{p})}}{|N_k(\mathbf{p})|} \quad (4)$$

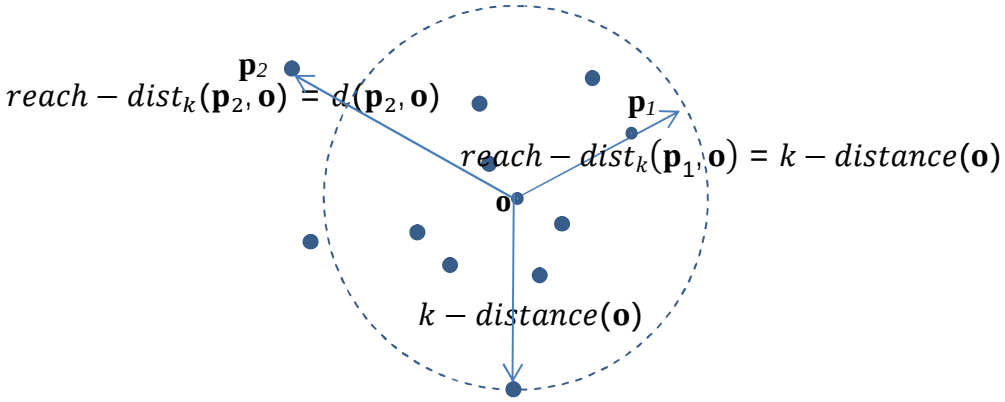


Fig.2: An illustration of *k-distance* ($\mathbf{o}$) and *reach-dist* $_k$ ($\mathbf{p}$,$\mathbf{o}$).

The LOF algorithm is extensively computational when working with datasets having large numbers of objects. To improve results of the LOF algorithm when working with low density patterns, connectivity-based outlier factor (COF) was proposed [6]. The COF of an object $\mathbf{p}$ is defined as ratio of its average chaining distance with average chaining distances of its k nearest neighbors. A distance between two disjoined-subsets $\mathbf{P}$ and $\mathbf{Q}$ of $\mathbf{X}$ is defined as the minimum distance between object $\mathbf{x} \in \mathbf{P}$ and $\mathbf{y} \in \mathbf{Q}$. A set based nearest path, *SBN-path*, from $\mathbf{p}_1$ of length r is a sequence $<\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_r>$ in such a way that $\mathbf{p}_{i+1}$ is the nearest neighbor of the set $\{\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_i\}$ with $1 \leq i \leq r$-1. A set based nearest trail, *SBN-*

trail, with respect to the *SBN-path* $<\mathbf{p}_1, \mathbf{p}_2, ..., \mathbf{p}_r>$ is defined as an ordered collection of r-1 edges $<\mathbf{e}_1, \mathbf{e}_2, ..., \mathbf{e}_{r-1}>$ with $1 \le i \le r$-1 and $\mathbf{e}_i=(\mathbf{o}_i, \mathbf{p}_{i+1})$ where $\mathbf{o}_i \in \{\mathbf{p}_1, \mathbf{p}_2, ..., \mathbf{p}_i\}$. The average chaining distance from an object $\mathbf{p}$ to $\mathbf{Q}\backslash\{\mathbf{p}\}$ is defined by (5) and the connectivity-based outlier factor of an object $\mathbf{p}$ with respect to its k neighbors is defined by (6).

$$ac-dist_Q(p) = \frac{1}{r-1}\sum_{i=1}^{r-1}\frac{2(r-i)}{r}dist(e_i) \quad (5)$$

$$COF_k(\mathbf{p}) = \frac{|N_k(\mathbf{p})|ac-dist_{N_k(\mathbf{p})}(\mathbf{p})}{\sum_{\mathbf{o}\in N_k(\mathbf{p})} ac-dist_{N_k(\mathbf{o})}(\mathbf{o})} \quad (6)$$

Similar to using LOF, when connectivity-based outlier factor values of all objects are gained, objects are ordered in a descending order based on the magnitudes of connectivity-based outlier factor values. Objects having the highest connectivity-based outlier factor are considered as outliers. Using COF might be better than using LOF in the case of provided datasets containing low density patterns. Nevertheless, using COF is more computational than using LOF. Thus, the COF and LOF algorithms are inappropriate to work with big datasets.

To be able to achieve more accurate outliers in more complex situations, influential measure of outliers (INFLO) was proposed [7]. Unlike LOF and COF algorithms, INFLO algorithm uses reverse nearest relationships to measure outlier degrees of objects. Influential measure of an object $\mathbf{p}$ is defined as ratio of density in its neighborhood with average density of objects in its reverse nearest neighbors. Density of an object $\mathbf{p}$, $den(\mathbf{p})$, is defined as reverse of k-$distance(\mathbf{p})$. The reversed nearest neighborhood of $\mathbf{p}$ is defined by (7). The *k-influential* space for $\mathbf{p}$ is defined by (8). Influential outlier score of an object $\mathbf{p}$ is defined by (9).

$$\Omega_R(\mathbf{p}) = \{\mathbf{q}\in\mathbf{X}|\mathbf{p}\in N_k(\mathbf{q})\} \quad (7)$$

$$IS_k(\mathbf{p}) = \Omega_R(\mathbf{p}) \cup N_k(\mathbf{p}) \quad (8)$$

$$INFLO_k(\mathbf{p}) = \frac{1}{den(\mathbf{p})}\frac{\sum_{\mathbf{o}\in IS_k(\mathbf{p})} den(\mathbf{o})}{|IS_k(\mathbf{p})|} \quad (9)$$

The INFLO algorithm is also an extensively computational consuming algorithm and inadequate to work with large datasets. Similar to the INFLO algorithm, the rank-based outlier detection algorithm (RBDA) [8] uses the concept of the reverse neighborhood to rank objects. Initially, rank of each object among its neighbors is computed. Secondly, outlier degrees of objects are calculated. Outlier degree of $\mathbf{p}$ is defined as ratio of sum of ranks of $\mathbf{p}$ among its neighbors with the number of its neighbors. Next, outlier degree values of objects are normalized based on these values and sizes of outliers are then detected. k objects having normalized outlier degrees which are greater than L are considered as k outliers. ODMR and ODMRD are two modified rank-based algorithms by assigning weights to clusters [11]. The RBDA, ODMR and ODMRD algorithms resemble aforementioned algorithm in terms of complexity, which are inappropriate to work with datasets having a large number of objects.

To provide a highly accurate result, a precise ranking method was proposed [10]. This is a multi-sampling method. At step i, $(1 \le i \le S)$, firstly, a subset having M objects is arbitrarily extracted from a provided dataset. Then, outlier scores of objects belonging to the subset are calculated. After finishing S steps, the final outlier score of each object is computed as the total sum of its outlier scores from S steps. Finally, objects are ranked based on the sum of outlier factor scores. Objects with the highest scores are considered as outliers. It is reported that this method can provide highly accurate results. However its execution time is tremendously long when working with big datasets due to the size of subsets, the number of subsets S can also be large numbers.

To resolve the weakness of using a single depth k of the previous algorithms and make use of the reversed nearest relationships to offer higher accuracy, a ranking-based outlier detection algorithm with various depths (RDOS) was proposed [11]. Unlike the LOF algorithm, the RDOS algorithm utilized both *k-distance* neighborhood and reversed *k-distance* neighborhood with various values of k to find the optimal k'. When obtaining an optimal value k', outlier scores of objects are calculated based on this k', and objects are ranked in a descending order based on scores. It was reported that the RDOS algorithm produces higher accuracy compared to the LOF, COF, INFLO, RBDA, and ORMRD algorithms. However, the main disadvantage of the RDOS algorithm is its execution time which is still too long compared to the LOF, COF, INFLO, RBDA, and ORMRD algorithms.

3 A Fast Outlier Detection Algorithm for Big Datasets

This algorithm is proposed not only to utilize the strength of the RDOS algorithm [11] but also to minimize its weakness. This proposed algorithm can produce almost the same results compared to the RDOS algorithm and can reduce a large part of the execution time when working with datasets having large numbers of objects. The proposed fast cell-various depths ranking-based outlier detection algorithm (Cell-RDOS) is comprised of three steps. The first step is to transform a big dataset to a small set of weighted cells using a proposed cell-based algorithm. The second step is to modify and apply the ranking-based algorithm with various depths to the weighted cell set to calculate outlier degrees of cells. The last step is to retrieve outliers' ids from ranked cells and remove outliers from the provided datasets. The overall solution of the proposed Cell-DROS is depicted in Fig.3.

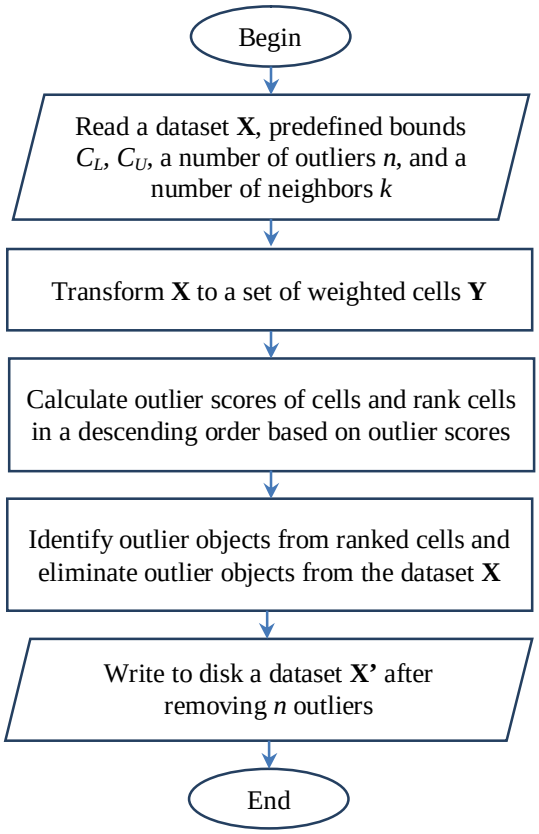


Fig.3: An illustration the proposed Cell-DROS solution.

3.1 A Proposed Cell-based Algorithm

This algorithm is designed to transform a dataset having a great number of objects to a significantly small set of weighted cells based on predefined lower and upper sizes of a desired set of weighted cells to reduce a large number of calculations.

Let $\mathbf{X}=(\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N)$ be a provided dataset having N objects and D attributes, $\mathbf{v}_{min}=(v_{min\,1}, v_{min\,2}, ..., v_{min\,D})$ and $\mathbf{v}_{max}=(v_{max\,1}, v_{max\,2}, ..., v_{max\,D})$ be the minimum and maximum values of attributes. Each domain $[v_{min\,j}, v_{max\,j}]$ is equally divided into M intervals to establish vectors $\mathbf{w}=(w_1, w_2, ..., w_D)$ and $\mathbf{m}_j=(m_{j1}, m_{j2}, ..., m_{jM})$ with $j=1,...D$. Values of $v_{min\,j}$, $v_{max\,j}$, w_j, m_{jk} are calculated by (11) to (14), respectively.

$$\mathbf{X} = \begin{pmatrix} x_{11} & x_{12} & \dots & x_{1D} \\ x_{21} & x_{22} & \dots & x_{2D} \\ \dots & \dots & \dots & \dots \\ x_{N1} & x_{N2} & \dots & x_{ND} \end{pmatrix} \quad (10)$$

$$v_{min\,j} = \min(x_{ij}) \text{ with } i=1,..., N \text{ and } j=1,...,D \quad (11)$$

$$v_{max\,j} = \max(x_{ij}) \text{ with } i=1,..., N \text{ and } j=1,...,D \quad (12)$$

$$w_j = \frac{v_{max\,j} - v_{min\,j}}{M} \text{ with } j=1,...,D \quad (13)$$

$$m_{jk} = v_{min\,j} + j \times w_j \text{ with } j=1,...,D \text{ and } k=0,...,M \quad (14)$$

Let C_L and C_U be the minimum and maximum numbers of unempty cells of a desired set of weighted cells, an initial value of M is calculated by (15).

$$M = 1 + (C_L + C_U)^{1/D} \quad (15)$$

After obtaining an initial value of M, the provided dataset will be divided into a set of cells as in Fig.4. And, objects are mapped into cells by using Algorithm 1.

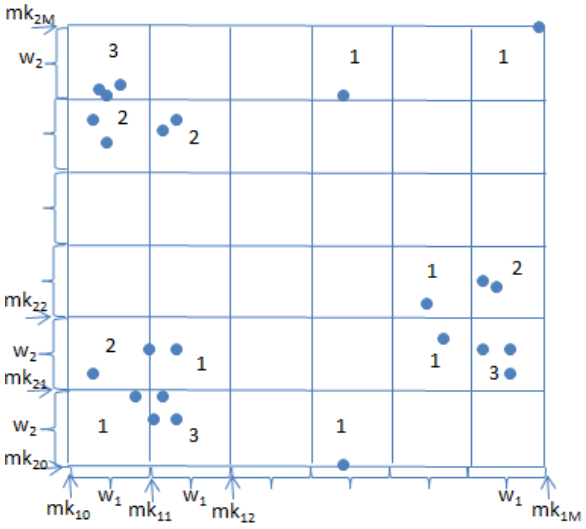


Fig.4: An illustration of dividing a dataset having D =2 into a set of cells.

Algorithm 1: Mapping (**X**, $\mathbf{m}_j$ with $j=1,...,D$)

Input: A dataset **X**, D marking vectors $\mathbf{m}_j$

Output: A copy of dataset **X** with new coordinates called **X'**

Begin

X'←**X**

For each data object $\mathbf{x'}_i \in \mathbf{X'}$ with $i=1,...,N$

For j from 1 to D

$stop \leftarrow 0$

$k \leftarrow 0$

While ($stop<1$) and ($k<M$)

If ($x'_{ij} \le m_{jk}$)

$x'_{ij} \leftarrow m_{jk}$

$stop \leftarrow 1$

Else

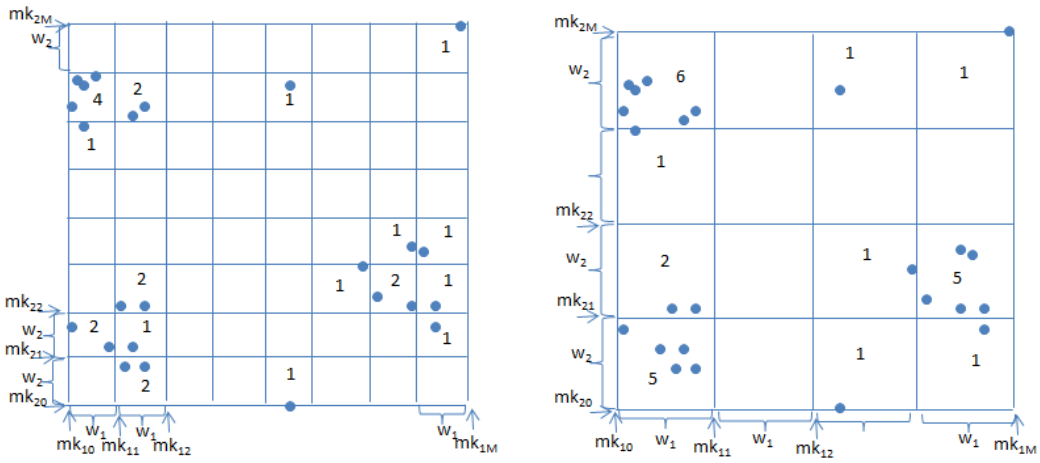
```
                    k←k+1
                End if
            End while
        End for
    End for
End
```

Due to the fact that the number of unempty cells created by the initial value M from (16) could be less than the lower bound C_L or greater than the upper bound C_U. The current value of M needs to be adjusted to get more acceptable cell sets. Let C be a number of cells created by the current value of M, U be a current number of unempty cells among C cells, Δ_C be difference between predefined lower bound and the current number of unempty cells calculated by (16), Δ_M be difference between an expected value of M and the current value of M calculated by (17). Values of M will be adjusted by (18) until obtaining an expected value of M which satisfies (19) or (20).

$$\Delta_C = |U - C_L| \tag{16}$$

$$\Delta_M = 1 + \Delta_C^{1/D} \tag{17}$$

$$M = \begin{cases} M + \Delta_M, & \text{if } U < C_L \\ M - \Delta_M, & \text{if } U > C_U \end{cases} \tag{18}$$



a) $M = M + \Delta_M$ b) $M = M - \Delta_M$

Fig.5: An illustration of adjusting values of M to increase, decease values of U.

Let M^-, M^+ be values of M at the previous and the next iterations, the process of finding a desired value of M will stop when values of U, M^-, M^+ satisfy (19). The left part of this equation means the number of unempty cells satisfies the predefined bounds. On the other hand, the right part means the process reaches an unstopping situation.

$$(C_L \leq U \leq C_U) \text{ or } (M^- = M^+) \quad (19)$$

In a specific case, the value of U is unchanged when values of M are changed a number of times. The process of finding an expected value of M should stop even though $U<C_L$ or $U > C_U$. Let n_{max} be a predefined maximum number of iterations the value of U is unchanged but values of M are changed, and n_{loop} be a number of times successive values of M are changed but the value of U is unchanged. The process of finding a desired value of M will stop when values of n_{max} and n_{loop} satisfy (20).

$$(n_{loop} = n_{max}) \quad (20)$$

Let $\mathbf{Y}=(\mathbf{y}_1, \mathbf{y}_2,...,\mathbf{y}_U)$ be a set of unempty weighted cells obtained from the provided dataset, $\mathbf{y}_i=(y_{i1}, y_{i2}, y_{iD}, w_i)$ be an unempty cell where y_{ij} is coordinates of cell $\mathbf{y}_i$, and w_i is weight of cell $\mathbf{y}_i$. Weight of a cell is defined as a number of objects belonging to that cell, and w_i is calculated by counting the number of objects $\mathbf{x}'_i \in \mathbf{X}'$ having the same coordinates. The proposed transformation step is depicted by Algorithm 2.

Algorithm 2: Cell-based_Transformation ($\mathbf{X}$, C_L, C_U)

Input: A dataset $\mathbf{X}$, predefined lower and upper bound called C_L, C_U

Output: A set of unempty cells called $\mathbf{Y}$

Begin

1. Calculate vectors $\mathbf{v}_{min}$ and $\mathbf{v}_{max}$ using (11) and (12), respectively.
2. Calculate an initial value of M using (15).
3. Calculate $\mathbf{w}$ and $\mathbf{m}_j$ with $j=1,...,D$ using (13) and (14), respectively.
4. Map data objects $\mathbf{x}_i = (x_{i1}, x_{i2}, ..., x_{iD})$ with $i=1,...,N$ to cells based on the value of $\mathbf{x}_i$ and the marking vectors $\mathbf{m}_j$ with $j=1,...,D$ using Algorithm 1.
5. Calculate a number of unempty cells called U.
6. Check stopping conditions (19), (20)
   - If (19) or (20) is satisfied, return a set of unempty cells $\mathbf{Y}$ and stop.
   - Otherwise, calculate Δ_C using (16) and Δ_M using (17), adjust M using (18), and repeat from step 3.

End

3.2 A Revised Ranking-based Algorithm with Various Depths

To be able to identify outlier objects from weighted cells and eliminate outliers from a provided dataset, cells must be calculated to discover the outlier degrees and then ranked according to these degrees. The ranking-based outlier detection

algorithm with various depths is modified and applied to this process. In this revised algorithm:

$w(\mathbf{p})$ is weight of cell $\mathbf{p}$,

$d(\mathbf{p}, \mathbf{q})$ is distance from cell $\mathbf{p}$ to cell $\mathbf{q}$,

k-distance$(\mathbf{p})$ is distance from cell $\mathbf{p}$ to its k^{th} nearest neighbor,

$N_k(\mathbf{p})$ is a set of k nearest neighbors of cell $\mathbf{p}$,

$d_k(\mathbf{p}, \mathbf{q})$ is a set of distances from cell $\mathbf{p}$ to cells $\mathbf{q} \in N_k(\mathbf{p})$,

$\Omega_k(\mathbf{p})$ is forward density around cell $\mathbf{p}$ at *k-distance*$(\mathbf{p})$,

$R_k(\mathbf{q})$ is reverse ranking of cell $\mathbf{p}$ with respect to cells $\mathbf{q} \in N_k(\mathbf{p})$,

$\Omega_R(\mathbf{q})$ is reverse density around cell $\mathbf{q}$ at *k-distance*$(\mathbf{p})$,

σ, $h_{optimal}$ and Ω_{smooth} are calculated using (21), (22) and (23), respectively [11]. The proposed algorithm is depicted as Algorithm 3.

$$\sigma = \frac{median(|\Omega_R - median(\Omega_R)|)}{0.6745} \tag{21}$$

$$h_{optimal} = \frac{0.9\sigma}{N^5} \tag{22}$$

$$\Omega_{\text{Smooth}} = \frac{1}{N h_{optimal}} \sum_{i=1}^{k} \frac{\exp\left(-\frac{1}{2}\left(\frac{\Omega_R - \Omega_R^i}{h_{optimal}}\right)^2\right)}{\sqrt{2\pi}} \tag{23}$$

Algorithm 3: Weighted-Cell Ranking($\mathbf{Y}$, k)

Input: Unempty cells $\mathbf{Y}$, an initial value k

Output: Ids of cells associated with outlier scores, and an optimal k '

Begin

1. *Find sorted distances associated with cells from each cell to its k nearest cells.*

   For each cell $\mathbf{p} \in \mathbf{Y}$

   Find *k-distance*$(\mathbf{p})$

   $d_k(\mathbf{p},\mathbf{q}) \leftarrow$ ascending sort$(d(\mathbf{p}, \mathbf{q}))$ with $\mathbf{q} \in N_k(\mathbf{p})$

   $N_k(\mathbf{p}) \leftarrow$ ascending sort$(N_k(\mathbf{p}))$ by $d(\mathbf{p}, \mathbf{q})$ with $\mathbf{q} \in N_k(\mathbf{p})$

   End for

2. *Find reverse ranks and reverse density of cells with various depths.*

   For each cell $\mathbf{p} \in \mathbf{Y}$

   For depth i from 2 to k

   For each $\mathbf{q} \in N_k(\mathbf{p})$

   $R_i(\mathbf{q}) \leftarrow$ reverse rank $\mathbf{p}$ by $\mathbf{q}$

End for

End for

$\Omega'_R(\mathbf{p}) \leftarrow \text{median}(R_i(\mathbf{q})/d_i(\mathbf{p},\mathbf{q}))$

End for

3. *Find median of reverse density*.

$\Omega'(\mathbf{p}) \leftarrow \text{median}(\Omega'_R(\mathbf{p}))$

4. *Find optimal k'*.

$$\sigma \leftarrow \frac{median(|\Omega'_R - median(\Omega'_R(\mathbf{q}))|)}{0.6745}$$

$h_{optimal} \leftarrow 0.9\sigma / N^5$

$\Omega_{max} \leftarrow 0$

$k_{optimal} \leftarrow 0$

For depth *i* from 2 to *k*

$$\Omega_{smooth} \leftarrow \frac{1}{N\, h_{optimal}\sqrt{2\pi}} \sum_{i=1}^{N} e^{-\frac{1}{2}\left(\frac{(\Omega'_R - \Omega'_R(\mathbf{q}))^2}{h_{optimal}}\right)}$$

If $\Omega_{smooth} > \Omega_{max}$ then

$\Omega_{max} \leftarrow \Omega_{smooth}$

$k_{optimal} \leftarrow i$

End if

End for

5. *Calculate outlier score for each cell*.

For each cell $\mathbf{p} \in \mathbf{Y}$

For depth *i* from 2 to *k*

$\Omega_i(\mathbf{p}) \leftarrow i/d_i(\mathbf{p},\mathbf{q})$

$score_i(\mathbf{p}) \leftarrow \frac{R_i - i}{\Omega_i(\mathbf{p}) \times w(\mathbf{p})}$

It means that a cell containing fewer objects has larger outlier degree compared to a cell containing more objects

End for

$\text{Score}(\mathbf{p}) \leftarrow \text{median}(\text{score}_i(\mathbf{p}))$

End for

6. *Sort cells in a descending order of scores*

$\mathbf{Y} \leftarrow$ descending sort($\mathbf{Y}$) by score ($\mathbf{p}$) with $\mathbf{p} \in \mathbf{Y}$

Return $\mathbf{Y}$ and $k_{optimal}$

End

The original outlier degree calculation equation with various depths is modified in such a way that a cell containing fewer objects has larger outlier degree compared to a cell containing more objects. Thus, real outliers are likely to be detected correctly.

Outputs of Algorithm 3 are weighted cells which are ranked in a descending order based on the outlier degree values. Cells containing more objects are likely to be ranked at lower positions because they have small values of outlier degrees. In other words, real outlier data objects are likely to be contained in the top ranked cells which contain fewer objects compared to the lower ranked cells.

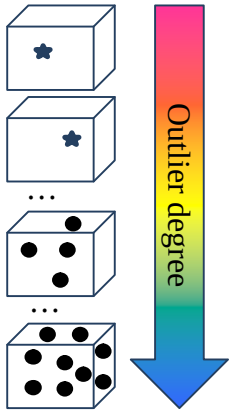


Fig.6: An example of outputs of Algorithm 3.

3.3 Outlier Identification and Elimination

By applying Algorithm 3, n expected outliers are more likely to be enclosed in the top m ranked cells ($m \leq n$). The purpose of this step is to identify ids of outliers contained in top ranked cells and eliminate n outlier data objects from the provided datasets. Let $\mathbf{Z} = (z_1, z_2, ..., z_n)$ be a list of n outliers, $\mathbf{Y}$ is an output of the previous step. The process of retrieving ids of outlier data objects from ranked cells and eliminate outliers from the provided dataset is depicted as Algorithm 4.

Algorithm 4: Outlier Retrieving And Elimination($\mathbf{X}$, n, $\mathbf{Y}$)
Input: A dataset $\mathbf{X}$, a number of outliers n, a set of ranked cells $\mathbf{Y}$.
Output: A set of data object as outliers $\mathbf{Z}$, dataset $\mathbf{X}$ after removing outliers
Begin
  $found \leftarrow 0$; $j \leftarrow 1$; $\mathbf{Z} \leftarrow \emptyset$
  while (($found < n$) and ($j \leq n$))
    For each object $\mathbf{x} \in \mathbf{X}$ and $\mathbf{x} \in \mathbf{y}_j$
      $\mathbf{Z} \leftarrow \mathbf{Z} + \{\mathbf{x}\}$
      $\mathbf{X} \leftarrow \mathbf{X} - \{\mathbf{x}\}$
      $found \leftarrow found + 1$
    End for
    $j \leftarrow j + 1$
  End while
  Return $\mathbf{Z}$ and $\mathbf{X}$
End

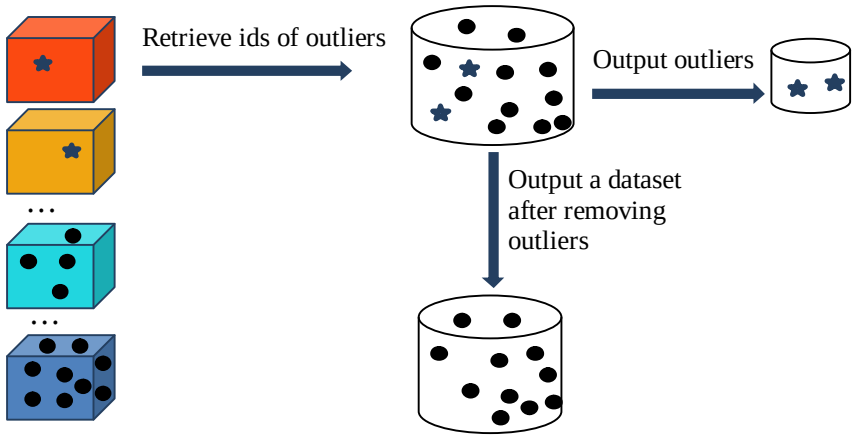


Fig.7: An Illustration of retriving ids of outliers from ranked cells and eliminating outliers from a provided dataset.

4 Experiments and Results

To compare results from the proposed Cell-RDOS algorithm with results from the previous RDOS method [11] in terms of matching detected outliers and execution time, algorithms were implemented using the C programming language, compiled by the TDM-GCC 4.8.1 64 bit release associated with the Dev C++ 5.9.2, running on a normal personal computer. The machine was configured with an Intel processor core i5-24000 CPU 3.10 GHz, 8GB of RAM, and Windows 7. A predefined k=10 was used as the number of nearest neighbors by both Cell-DROS and DROS algorithms. A predefined n_{max}=10 was used as a predefined maximum number of iterations the value of U is unchanged but values of M are changed. And, 29 pairs $<C_L, C_U = C_L+1{,}000>$ with $C_L = i \times 1{,}000$ and i=1,...,29 were tested by the proposed Cell-RDOS algorithm.

Datasets used in the experiments were collected from previous researches. They were a part of the *Activity Recognition* dataset and a part of the *Person Activity* dataset obtained from UCI website [13], a part of the *TDriveTrajectory* dataset obtained from the Microsoft website [14], the *2010_09_10_jam_+40* dataset obtained from the NASA website [15], and the *Simulation2D1* dataset obtained from [16] with 100 outliers added. Information of the selected datasets is shown in Table 2.

Table 2: Information of datasets.

Datasets	No. of objects	No. of attributes	No. Outliers need to be detected
Activity Recognition	162,501	3	10
Person Activity	164,860	3	10
TDrive Trajectory	176,424	2	7
2010_09_10_jam_+40	584760	6	10
Simulation2D1	800,059	2	100

Criteria used to compare the proposed Cell-DROS and the previous DROS algorithm are numbers of matching detected objects and the average of execution time.

Table 3: Comparison of matching results between two algorithms with C_L=29,000.

Datasets	No. of objects	No. of detected outliers	No. of matching detected objects	Percentage of matching
Activity Recognition	162,501	10	9	90%
Person Activity	164,860	10	9	90%
TDrive Trajectory	176,424	7	6	85%
2010_09_10_jam_+40	584,760	10	10	100%
Simulation2D1	800,059	100	100	100%

Table 4: Comparison of executing time between two algorithms with C_L=29,000

Datasets	No. of objects	Executing time (in minutes)		Reduced time
		RDOS	*Cell-RDOS*	
Activity Recognition	162,501	509.99	8.33	98.37%
Person Activity	164,860	533.93	4.46	99.16%
TDrive Trajectory	176,424	611.85	398.55	34.86%
2010_09_10_jam_+40	584,760	12,086.84	8.85	99.93%
Simulation2D1	800,059	9,807.62	12.05	99.88%

When working with datasets having fewer than 200,000 objects, results from Table 3 shows that unmatched detected outliers between the proposed Cell-DROS solution and the previous RDOS algorithm are only 1 object among the number of needed outliers. However, results in Table 4 shows that the proposed Cell-RDOS algorithm can reduce the execution time from 34.86% to 99.16% compared with the RDOS algorithm. Among three datasets having fewer than 200,000 objects, the execution time of the dataset *TDrive Trajectory* is only reduced by 34.86% because this dataset had low density patterns.

When working with bigger datasets having 584,760 and 800,059 objects, results in Table 3 shows that the proposed Cell-RDOS algorithm produces exactly the same detected outliers when compared to the results of the RDOS algorithm. However, results in Table 4 shows that the proposed Cell-RDOS algorithm can reduce up to 99.88% of the execution time compared to the RDOS algorithm.

In the case of the dataset *2010_09_10_jam_+40*, the previous DROS algorithm consumed 12,086.84 minutes, approximately equal to 8.39 days, to detect and remove 10 outliers from the provided dataset. However, the proposed Cell-DROS algorithm produced exactly the same results within only 8.85 minutes.

In the case of the dataset *Simulation2D1*, the previous RDOS algorithm consumed 9,807.62 minutes. Whereas, the proposed Cell-RDOS algorithm consumes only 12.05 minutes to detect the same outliers. This reduced time is approximately equal to 163.27 hours or 6.8 days.

Moreover, based on the selected datasets, experiment results also show that the proposed Cell-RDOS algorithm produced the same results when the lower bound C_L was assigned 27,000 cells, 28,000 cells, and 29,000 cells. And, numbers of matching detected outliers would become smaller when C_L obtains smaller values.

5 Conclusions and Discussions

The ranking-based outlier detection with the various depths algorithm (RDOS) is better than many other ranking-outlier detection algorithms in terms of outlier recognition including LOF, COF, INFLO, RBDA, and ORMRD algorithms. However, it is not appropriate to work with very large datasets due to its shockingly long execution time. This paper proposes a Cell-RDOS algorithm for outlier detection in big datasets. The proposed solution can utilize the merit of the RDOS algorithm and resolve its weakness which is long executing time.

Experiment results show that the proposed Cell-DROS algorithm satisfies the design goals. It produces the same results compared to the RDOS algorithm and minimizes a large part of the RDOS algorithm's execution time when analyzing large datasets.

The proposed Cell-RDOS algorithm can be considered as a great tool for ranking-based outlier detection for big datasets. Accuracy level of the proposed Cell-RDOS algorithm matches the accuracy level of the RDOS algorithm. However, the Cell-RDOS method can reduce up to 99% of execution time compared to the RDOS method when working with very large datasets such as *2010_09_10_jam_+40*, *Simulation2D1*. Based on the experiment results from the collected datasets, to gain a high accuracy level, the predefined lower bound should be at least 27,000 cells.

References

[1] Chandola, V., Banerjee, A. and Kumar, V. 2009. Anomaly detection: A survey, *ACM Computing Surveys*, vol.41, no.15, 1-15.

[2] Hawkins, D. M. 1980. Introduction, in *Identification of Outliers*, Chapman & Hall, 1-9.

[3] Aggarwal, C. C. 2015. Outlier Analysis, in *Data Mining*, Springer International Publishing Switzerland, 237-263.

[4] Shaikh, S. and Kitagawa, H. 2014. Top-k Outlier Detection from Uncertain Data, *International Journal of Automation and Computing*, vol.11, 128-142.

[5] Breunig, M.M., Kriegel, H.P., Raymond, T.Ng, and Sander, J. 2000. LOF: identifying density-based local outliers, *ACM SIGMOD Record*, vol.29, 93-104.

[6] Tang, J., Chen, Z., Fu, A.W.C. and Cheung, D.W.L. 2002. Enhancing Effectiveness of Outlier Detections for Low Density Patterns, in *the 6th Pacific-Asia Conf. on Advances in Knowledge Discovery and Data Mining*, 535-548.

[7] Jin, W., Tung, A.H., Han, J. and Wang, W. 2006. Ranking Outliers Using Symmetric Neighborhood Relationship, *Advances in Knowledge Discovery and Data Mining*, vol.3918, 577-593.

[8] Huang, H., Mehrotraa, K. and Mohana, C.K. 2013. Rank-based outlier detection, *Journal of Statistical Computation and Simulation*, vol.83, 518-531.

[9] Huang, H., Mehrotra, K. and Mohan, C. 2012. Algorithms for Detecting Outliers via Clustering and Ranks, *Advanced Research in Applied Artificial Intelligence*, vol.7345, 20-29.

[10]Ha, J., Seok, S. and Lee, J.S. 2015. A precise ranking method for outlier detection, *Information Sciences*, vol.324, 88-107.

[11]Bhattacharya, G., Ghosh, K. and Chowdhury, A.S. 2015. Outlier detection using neighborhood rank difference, *Pattern Recognition Letters*, vol.60, 24-31.

[12]Hodge, V.J. 2014. Outlier Detection in Big Data, in *Encyclopedia of Business Analytics and Optimization*, vol.5, 1762-1771.

[13]Lichman, M. Machine Learning Repository, http://archive.ics.uci.edu/ml/.

[14]Yuan, J., Zheng, Y., Xie, X. and Sun, G. 2011. Driving with knowledge from the physical world, in *the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, San Diego, California, USA, 316-324.

[15]Edward B. 2014. 2010_09_10_jam_+40 Lab.

https://c3.nasa.gov/dashlink/resources/?type=28

[16]Hieu, D.V. and Meesad, P. 2015. A Cell-MST-Based Method for Big Dataset Clustering on Limited Memory Computers, in *the 7th International Conference on Information Technology and Electrical Engineering*, Chiang Mai, Thailand, 632-637.