Turkish Journal of Electrical Engineering & Computer Sciences

http://journals.tubitak.gov.tr/elektrik/

Research Article

Turk J Elec Eng & Comp Sci
(2020) 28: 2657 – 2669
© TÜBİTAK
doi:10.3906/elk-1901-82

An efficient storage-optimizing tick data clustering model

Haleh AMINTOOSI[1]*, Masood NIAZI TORSHIZ[2], Yahya FORGHANI[2], Sara ALINEJAD[2]
[1]Department of Computer Engineering, Faculty of Engineering, Ferdowsi University of Mashhad, Mashhad, Iran
[2]Department of Computer Engineering, Faculty of Engineering, Mashhad Branch, Islamic Azad University, Mashhad, Iran

Received: 11.01.2019 • **Accepted/Published Online:** 27.04.2020 • **Final Version:** 25.09.2020

Abstract: Tick data is a large volume of data, related to a phenomenon such as stock market or weather change, with data values changing rapidly over time. An important issue is to store tick data table in a way that it occupies minimum storage space while at the same time it can provide fast execution of queries. In this paper, a mathematical model is proposed to partition tick data tables into clusters with the aim of minimizing the required storage space. The genetic algorithm is then used to solve the mathematical model which is indeed a clustering model. The proposed method has been evaluated on a real-world weather tick dataset and compared to the storage-optimizing hierarchical agglomerative clustering (SOHAC) algorithm. The experiments show that our proposed method substantially outperforms SOHAC in achieving smaller values for compression ratio while reducing the execution time for small number of clusters.

Key words: Tick data, compression, clustering

1. Introduction

Monitoring phenomena such as weather information, blood pressure, stock market trading, and traffic condition involves collecting and managing large volume data, known as big data [1–3]. An important characteristic of such data is rapid production or tick by tick change over time. Moreover, the way these data values fluctuate is generally of great importance for analysts. These characteristics result in a very large collection of data that changes very fast and thus need to be stored efficiently and retrieved quickly. Such data is often called tick data [4]. An example of tick data is the weather condition information that consists of different characteristics of the weather, such as temperature, humidity score, air pressure, wind speed, collected at regular time stamps (e.g., every minute). There are many studies on data compression [5] that are aimed at reducing the amount of storage required for such large volumes of data. However, as mentioned in [6], these methods are not able to guarantee a fast and efficient response to queries since they require a large archive to be decompressed in order for a query to be answered, which is computationally expensive and inefficient. Thus, proposing a method that makes a good tradeoff between the data compression and quick access to the content of compressed data is a challenge. In order to address this challenge, authors in [6] proposed a novel clustering algorithm for tick data, called storage-optimizing hierarchical agglomerative clustering (SOHAC), that vertically partitions a tick data into a number of clusters. The clustering is done in a way to minimize the amount of required storage space and at the same time, efficiently search and service analytic queries. Later, they improved their method and proposed QuickSOHAC [7] which achieved substantial clustering speedup without significant loss of clustering

*Correspondence: amintoosi@um.ac.ir

quality. The main idea behind their work is to cluster the tick data matrix vertically in a way that rows of each cluster change together over time. In this case, if n subsequent rows contain the same values in all the columns within a cluster, one row is kept and other n-1 subsequent rows that have similar data are omitted, thus leading to better compression. Table 1 demonstrates a tick data table which contains the features of weather condition monitored by a weather station, and Table 2 shows its clustered data.

Table 1. A weather information tick data table.

Time	Temp. (°C)	Hum. (%)	Press. (Pa)	Wind (v) (km/h)	Wind (dir.)	Radiation	Outlook
10:21	15	20	100 200	5	SW	Low	Cloudy
10:22	16	20	100 200	5	SW	Low	Cloudy
10:38	16	30	100 100	5	SW	Low	Cloudy
10:40	17	30	100 100	5	SW	Medium	Sunny
10:43	18	30	100 100	10	SW	Medium	Sunny
10:44	18	30	100 100	15	W	Medium	Sunny
10:51	18	20	100 200	15	W	Medium	Sunny

Table 2. The clustered weather data.

Time	Hum. (%)	Press. (Pa)
10:21	20	100 200
10:38	30	100 100
10:51	20	100 200

Time	Temp (°C)	Wind (v) (km/h)	Wind (dir.)	Radiation	Outlook
10:21	15	5	SW	Low	Cloudy
10:22	16	5	SW	Low	Cloudy
10:38	16	5	SW	Low	Cloudy
10:40	17	5	SW	Medium	Sunny
10:43	18	10	SW	Medium	Sunny
10:44	18	15	W	Medium	Sunny
10:51	18	15	W	Medium	Sunny

Despite outperforming some other clustering algorithms [8], SOHAC has some limitations: First, it takes the assumption that all the data cells in the tick data matrix have the same size and need the same physical storage. This assumption, however, is not true for most large-scale tick datasets that store different kinds of attributes with different required storage space about a phenomenon. Taking the nonuniform storage space requirement of the cells into account leads to a more realistic compression model with better compression rate. Second, SOHAC is a heuristic algorithm [9] which finds a good suboptimal clustering for tick data. In other words, not all possible clusters but some are considered in order to find the one with the best storage space (best compression ratio). Specifically, in each iteration of SOHAC, all possible 2-combinations of columns or column subgroups (and not 3-combinations of columns and their subgroups for example) are taken into account as potential clusters. This process, however, is still computationally expensive especially for large scale datasets. In order to address the abovementioned issues, in this paper, a mathematical model is proposed for clustering tick data with the aim of minimizing the storage space. Then, the genetic algorithm [10, 11] is used to solve the proposed model. In our proposed model, we take into account the size of columns. Therefore, our proposed model can reach a smaller compression ratio (i.e. better performance) compared with SOHAC and quickSOHAC while at the same time, reduce the execution time for small number of clusters. Besides, our proposed mathematical model has less time complexity, as compared with SOHAC. Similar to methods such as SOHAC and quickSOHAC, our proposed model is only efficient for relatively stationary data in which no sudden changes are observed most of the time such as weather data. Different approaches must be used for data with quite different dynamics.

The remainder of the paper is as follows: Related work section reviews the studies concentrating on tick data management. The preliminaries section expresses the preliminaries and basic concepts referenced throughout the paper. The detailed description of our proposed approach is explained next. Simulation results and the validation of the proposed approach as well as a comparative study are presented in the evaluation section. Finally, the conclusion section provides discussion of the overall coverage of the article and concluding remarks

2. Related work

Similar to weather condition data, there exists high-resolution data related to the transactions on stock market [4], risk analysis [12], and currency exchange rates [13, 14], known as tick data. There has been considerable work focusing on statistical properties [13] and analysis of financial tick data [14]. However, few [15] have investigated the challenges related to the storage of high dimensional tick data, which is a core component of such analytic systems. Authors in [16] proposed a solution to summarize tick data time series by means of using multiscale wavelet analysis to separate out the trend, cyclical fluctuations, and autocorrelational effects. However, storing the summary of tick data table is not desired as compared to storing the entire tick data table. Regular compression methods [5], though suitable for efficiently storing large archives of data, are computationally expensive due to the need for data decompression per each query execution. In other words, conventional clustering algorithms will lead to suboptimal decompositions since they are not originally designed for tick data storage minimization. Thus, there is a need for efficient clustering and compression techniques for tick data in order to address the storage challenge and quickly respond to the queries [7]. The most related work that concentrates on supporting the storage and compression of tick data was first presented in [17], where a new clustering solution called SOPAC was proposed. Its main idea was based on the search for a partitional clustering that optimizes storage space. However, their approach did not guarantee an optimal solution, but was able to reduce the storage space substantially for real-world datasets compared to conventional partitional clustering algorithms. The authors then extended their previous work and proposed SOHAC [6], which was a new clustering algorithm that minimized storage space required for a tick data matrix. SOHAC was built on the hierarchical agglomerative strategy. At the beginning, each column is considered a separate cluster. Then, clusters are iteratively merged together as long as the current number of clusters is more than k, the user-defined number of partitions. Two clusters are merged if their combination leads to minimal storage size of the decomposed matrix. The proposed clustering approach was compared against different clustering algorithms such as k-means, single linkage, and average linkage on real-world tick datasets and showed substantially better performance in terms of achieving smaller compression ratio. A more detailed description of SOHAC will be presented in the following section. Later, the authors aimed at speeding up SOHAC and proposed a new lower bounding algorithm, called QuickSOHAC [7], which is more applicable to high dimensional tick data. As mentioned above about SOHAC, those two clusters are merged, which leads to minimal storage size. Instead of calculating the exact required storage size, the lower bound storage is considered. In that case, if two clusters (C_i) and (C_j) are considered to be merged and if there are other pairs of clusters whose required storage space is exactly s, it is unnecessary to calculate the exact storage size resulting from merging the clusters (C_i) and (C_j) if we know that merging (C_i) and (C_j) results in a storage size of $(s_1) > (s)$. The experimental results showed that the proposed lower bounding algorithm QuickSOHAC speeds up the clustering process by a factor of 4–6. Although SOHAC and QuickSOHAC have been shown to be able to specifically address the challenges of tick data storage, they have some limitations. First, in both methods, it is assumed that all the data cells in the tick data matrix are in the same size and hence occupy the same physical storage. However, for most large-scale tick datasets that store different kinds of characteristics with different required storage space about a phenomenon, such assumption is not true. Taking the nonuniform storage space requirement of the cells into consideration is believed to result in a more realistic compression model. It is also expected that it leads to smaller values for compression ratio. Second, SOHAC is based on a heuristic algorithm which considers some potential clusters (and not all possible ones) in order to find the one with the best storage space (lowest compression ratio). This will result in a suboptimal clustering for tick data. This process, however, is still computationally expensive especially for large scale datasets. In order to address the abovementioned challenges, we propose a mathematical model that is able to minimize the storage space while at the same time compressing the data in an efficient and timely manner.

3. Preliminaries

In this section, first, the basic idea of SOHAC and quickSOHAC is explained and then, the compression ratio parameter is described. Table 3 presents a brief description of the notations used within the rest of the paper.

Table 3. Notation description.

Notation	Description
n	Number of instances of phenomenon observation
m	Number of attributes describing the phenomenon
k	Number of clusters
c	Total number of cells in all clusters
A	The set of attributes, A=$\{a_1, a_2, ..., a_m\}$
C	The set of clusters, C=$\{C_1, C_2, ..., C_k\}$
w_i	The weight of attribute a_i which equals its size
r_i	Number of rows in C_i
CS	Cluster space: the space required to store clusters C_1, C_2, ..., C_k
M	The $n \times m$ tick data matrix
I	The $n \times m$ change indicator matrix showing change/no change in a cell value of M, $I(i,j) \in \{0,1\}$
U	The $k \times m$ clustering matrix showing whether attribute a_j resides in cluster C_i or not

3.1. SOHAC and QuickSOHAC

The tick data is represented by a $n \times m$ table denoted by M where n is the number of observations of a phenomenon at different time stamps, and m is the number of attributes $(a_1, ..., a_m)$ describing the phenomenon at a time stamp. One of the attributes is called the index attribute which can be an ascending integer value used to index the rows of the table. The related column of index attribute is called the index column. Timestamp is a common example of an index attribute. Other columns of tick dataset are called regular columns. SOHAC partitions regular columns of the table M vertically into k disjoint partitions. Once the partitioning is done, the index column is added to each partition (for the sake of reconstruction of the original M) to form the clusters

$C_1, C_2, , C_k$. As it was mentioned, Tables 1 and 2 demonstrate a tick data table which contains the features of weather condition monitored by a weather station, and the result of clustering tick data vertically into two clusters, respectively. As can be seen, in each cluster, the records which are the same as the previous ones (excluding the time stamp column) can be deleted without losing any information about the weather condition. Strictly speaking, if such records are deleted, the original tick data can be reconstructed in a lossless manner, because it is supposed that the weather condition between two time stamps in a partition does not change. SOHAC partitions regular columns of M vertically such that the overall storage needed to store the clusters $C_1, C_2, , C_k$ is as less as possible. SOHAC uses a heuristic algorithm to find a good suboptimal clustering. To do so, in each iteration of SOHAC, all possible 2-combinations of columns or subgroups of columns are taken into account as potential clusters in order to find the one with the best storage space (best compression ratio). This process, however, is suboptimal and also computationally expensive especially for large-scale datasets.

QuickSOHAC is aimed at speeding up SOHAC, so it ignores the clustering whose storage size lower bound is greater than the best clustering storage size found so far. Computing storage size lower bound of a clustering is less time-consuming than computing storage size of that clustering. Storage size lower bound of a cluster constructed by merging two clusters C_1 and C_2 with n_1 and n_2 columns, respectively, is as follows: $\frac{1}{2}(n_1 + n_2)StorageSize(mfcc(C_1), mfcc(C_2))$ where mfcc(C) denotes the most frequently changing column of cluster C. Therefore, $StorageSize(mfcc(C_1), mfcc(C_2))$ computes the storage size of a cluster with only two columns which is less time-consuming than computing the storage size of the original cluster with $n_1 + n_2$ columns.

3.2. Compression ratio

An effective parameter for evaluating the effectiveness of the clustering and compression is the compression ratio. The compression ratio (CR), is the ratio of the storage space needed to store the original tick data table to the storage space required to store the clusters. In [6, 7], authors assume that all data cell values (i.e. attribute values) have the same size and hence measure the storage space of each cluster by counting the number of cells within that cluster (The same has been done for measuring the storage space of original data table). In this article, we aim at adhering to real situations where attributes may vary in size. Therefore, we consider a weight for each data cell (attribute) which equals the size of the cell. In other words, CR is expressed by Equation 1.

$$CR = \frac{\sum_{1 \le i \le k}(r_i \sum_{a_j \in C_i} w_j)}{n \sum_{1 \le j \le m} w_j}, \tag{1}$$

where r_i is the number of distinct rows in i_{th} cluster.

3.3. Genetic algorithm

Genetic algorithm (GA) is a metaheuristic inspired by the process of natural selection that belongs to the larger class of evolutionary algorithms (EA). Genetic algorithms are commonly used to generate high-quality solutions to optimization and search problems by relying on bio-inspired operators such as mutation, crossover, and selection.[1] John Holland introduced genetic algorithms in 1960 based on the concept of Darwin's theory of evolution; afterwards, his student David E. Goldberg extended GA in 1989 [2]. Genetic algorithm is started with a set of solutions (represented by chromosomes) called population. Solutions from one population are taken and used to form a new population. This is motivated by a hope that the new population will be better than the old one. Solutions which are selected to form new solutions (offspring) are selected according to their fitness; the more suitable they are, the more chances they have to reproduce. Algorithm 1 is the detail of basic genetic algorithm.

Algorithm 1. Outline of the basic genetic algorithm.

```
1. [Start] Generate random population of n chromosomes (suitable solutions for the problem)
2. [Fitness] Evaluate the fitness f(x) of each chromosome x in the population
3. [New population] Create a new population by repeating following steps until the new population is complete
    (a) [Selection] Select two parent chromosomes from a population according to their fitness (the better fitness, the bigger chance to be selected)
    (b) [Crossover] With a crossover probability cross over the parents to form a new offspring (children). If no crossover was performed, offspring is an exact copy of parents.
    (c) [Mutation] With a mutation probability mutate new offspring at each locus (position in chromosome).
    (d) [Accepting] Place new offspring in a new population
4. [Replace] Use new generated population for a further run of algorithm
5. [Test] If the end condition is satisfied, stop, and return the best solution in current population
6. [Loop] Go to step 2
```

4. Our proposed model

As mentioned earlier, the aim of clustering is to have partitions with regular columns whose rows change as less times as possible, because those records of a partition which do not change can be removed without information loss. For this purpose, a binary indicator matrix I is derived from the original tick data table, as expressed by Equation 2:

$$I(i,j) = \begin{cases} M(i,j) & \text{if } j_{th} \text{ column is an index column} \\ 0 & \text{if i>1 and } M(i,j) = M(i-1,j) \\ 1 & \text{otherwise} \end{cases} \tag{2}$$

where M is tick data table, and M(i,j) is j-th column of i-th row of M. As equation 2 illustrates, when the value of a cell equals the value of the cell in the same regular column of the previous row of the table M, the entry belonging to that cell in the change indicator matrix I is set to zero; otherwise, it is set to 1. The change indicator matrix expresses which groups of columns change as less times as possible and hence can be put in the same cluster. Table 4 depicts a tick data related to the weather condition information as well as its change indicator matrix. Here, “Time” column is the index column.

Clustering is a NP-Hard problem. As mentioned above, SOHAC and quickSOHAC use heuristic algorithm to find a good suboptimal clustering. In other words, only some of all possible clusters are considered for selecting

Table 4. A weather information tick data table and its change indicator.

Time	Temp. (°C)	Hum. (%)	Press. (Pa)	Wind (v) (km/h)	Wind (dir.)	Radiation	Outlook
10:21	15	20	100 200	5	SW	Low	Cloudy
10:22	16	20	100 200	5	SW	Low	Cloudy
10:38	16	30	100 100	5	SW	Low	Cloudy
10:40	17	30	100 100	5	SW	Medium	Sunny
10:43	18	30	100 100	10	SW	Medium	Sunny
10:44	18	30	100 100	15	W	Medium	Sunny
10:51	18	20	100 200	15	W	Medium	Sunny

(a) Weather information tick data table

Time	Temp. (°C)	Hum. (%)	Press. (Pa)	Wind (v) (km/h)	Wind (dir.)	Radiation	Outlook
1	1	1	1	1	1	1	1
1	0	0	0	0	0	0	0
0	1	1	1	0	0	0	0
1	0	0	0	0	0	1	1
1	0	0	0	1	0	0	0
0	0	0	0	1	1	0	0
0	1	1	1	0	1	0	0

(b) Change indicator

the best. To be more precise, in each iteration of SOHAC, all possible 2-combinations of columns or column subgroups are taken into account as potential clusters in order to find the one with the best storage space (best compression ratio). Other combinations of clusters are ignored. This process, however, is still computationally expensive especially for large-scale datasets. Therefore, in this paper, a metaheuristic algorithm is used to cluster tick data vertically. The clustering is done in a way to minimize the amount of required storage space and at the same time to allow search and analytic queries to be executed efficiently. The main idea is to cluster the tick data matrix vertically in a way that rows of each cluster change as less times as possible. In this case, if n subsequent rows contain the same values in all the columns within a cluster, one row is kept and other n-1 subsequent rows that have similar data are omitted, thus leading to better compression. The main difference between our proposed clustering model and SOHAC and QuickSOHAC is that we take into account the size of columns while SOHAC and QuickSOHAC do not. Therefore, our proposed model can reach a smaller compression ratio (i.e. better performance) compared with SOHAC and quickSOHAC. Our proposed model is described as in Equation 3.

$$\min_{u} \sum_{i=1}^{k} (NumberOfNonZeroVectors(I \bigodot U_i) \times \sum_{j=1}^{m} (U_{ij} \times w_j)), \text{ subject to} \begin{cases} \forall_j : \sum_{i=1}^{k} U_{ij} = 1 \\ \forall_{i,j} : U_{i,j} \in \{0,1\} \end{cases} \tag{3}$$

where U is a binary $k \times m$ matrix that expresses whether or not an attribute (i.e. column) has been allocated to a cluster. In other words, U is membership matrix which can be expressed by Equation 4:

$$U_{ij} = \begin{cases} 1 & \text{if } a_j \text{ is allocated to } C_i \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

where a_j denotes j_{th} column of tick data M, w_j is the size of j_{th} column a_j, C_i is i_{th} cluster, and

I is the indicator matrix of the tick data M. The constraint $\sum_{i=1}^{k} U_{ij} = 1$ guarantees that each attribute a_j is only allocated to one cluster. $\sum_{j=1}^{m}(U_{ij} \times w_j)$ computes the storage size of i_{th} cluster. The function NumberOfNonZeroVectors(.) counts the number of nonzero row vectors of its matrix argument. $(I \bigodot U_i)$ denotes element-wise multiplication of the matrix I and the row vector U_i is replicated to an appropriate matrix size. Therefore, $NumberOfNonZeroVectors(I \bigodot U_i)$ counts the number of nonzero row vectors of i_{th} cluster. As mentioned earlier, zero row vector of clusters are removed and no storage space is needed to store such rows. Thus, $(NumberOfNonZeroVectors(I \bigodot U_i) \times \sum_{j=1}^{m}(U_{ij} \times w_j))$ is the storage size of i_{th} cluster. Notice that the only variable of our proposed model is the binary matrix U. The average performance of any metaheuristic algorithms across all possible problems is identical [18]. A chromosome in the genetic algorithm which is used as a solution of a problem is a binary vector. Therefore, genetic algorithm is well suited for solving binary optimization problems such as our proposed model. Therefore, our proposed model can be solved efficiently using the genetic algorithm. Solving a model with binary variables using a metaheuristic algorithm such as genetic algorithm is more efficient than solving a model with real variables because in a model with binary variables, the number of different possible solutions is finite, contrary to a model with real variables. Notice that some clustering methods such as *k-means* compute cluster centers which are real variable and are not as efficient as our proposed clustering model.

Moreover, k-means algorithm has two sets of variables, i.e. membership values and cluster centers while our proposed model has only one set of variables, i.e. membership values. In other words, our proposed model has less variables than k-means. Therefore, solving our proposed model is less time-consuming than k-means.

As it was mentioned, the binary matrix U is the only variable of our proposed model introduced in Equation 3. We vectorized this matrix and then used basic genetic algorithm to obtain the optimal solution of our proposed model. Each chromosome is an instance of vectorized version of the matrix U. Genetic algorithm is started with a set of solutions (represented by chromosomes) called population. Therefore, we started with a set of random matrices (solutions of our proposed model) that each one satisfies the constraints of our model as initial population. Each matrix U or each solution determines a different clustering of data. Each matrix of population may be our optimal solution. The objective function of our proposed model which is the inverse of fitness function determines which matrix or which clustering is the best. Solutions from one population are taken and used to form a new population. This is motivated by a hope that the new population will be better than the old one. Solutions which are selected to form new solutions (offspring) are selected according to their fitness; the more suitable they are the more chances they have to reproduce. Algorithm 1 is the detail of basic genetic algorithm. We used single-point crossover and bitwise mutation in the genetic algorithm. Our proposed model clusters data such that data can be compressed much efficiently. Similar to SOHAC and quick SOHAC, our proposed model is not used for compression of future incoming data. Future incoming data must also be saved and then after a while, our proposed model must be optimized over the saved data in order to compress them.

4.1. Time complexity analysis

Experimental results show that the genetic algorithm is even faster than the heuristic algorithm used in SOHAC and QuickSOHAC. In each iteration of SOHAC and QuickSOHAC, only all possible 2-combinations of columns or column subgroups are taken into account as potential clusters in order to find the one with the best storage space (best compression ratio). These all possible 2-combinations of columns are $\binom{m}{2}$ at the first iteration. Computing compression ratio of each possible combination involves reading the whole of the table M with the order $O(n \times m)$. After each iteration, one of the columns or one of column subgroups of the table M is merged into another column or column subgroup. Therefore, at each iteration, the number of columns or column subgroups is decreased by one. This process is iterated until k subgroups remain. Therefore, the time complexity of SOHAC and QuickSOHAC is shown in Equation 5:

$$(\binom{m}{2}+\binom{m-1}{2}+...+\binom{k}{2})\times(n\times m)=(\frac{m\times\ (m-1)}{2}+\frac{m-1\times\ (m-2)}{2}+...+\frac{k\times(k-1)}{2})\times(n\times m) \quad (5)$$

Therefore, the time order of SOHAC and QuickSOHAC is $O(m^4 n)$ for $k=2$ which is computationally expensive specially for large-scale datasets. The number of computing compression ratio (fitness function) in the genetic algorithm is $k \times$ "*population size*" $\times$ "*no. of iteration*". Therefore, the time order of our proposed algorithm is $O(k \times$ "*population size*" $\times$ "*no. of iteration*" $\times n \times m)$. Therefore, if $(k \times$ "*population size*" $\times$ "*no. of iteration*") is much less than m^3, our proposed algorithm is computationally more efficient than SOHAC. This situation occurs especially when the number of columns of tick data, i.e. m, is large.

5. Evaluation

In this section, the performance of our proposed clustering model is evaluated. First, the experimentation setup along with the parameters considered for performance evaluation and the dataset used in experiments are explained. Then, the simulation results are presented.

5.1. Experiment setup

Our proposed model was solved using the basic genetic algorithm with tournament selection, single-point crossover and bitwise mutation, and a series of experiments were conducted to evaluate its performance. Best values of crossover and mutation rates depend on problem concerned. For complicated search spaces a crossover rate greater than 0.5 will help searching at the beginning. However, with progress it should be reduced to a value near 0.1 or 0.2. Mutation probabilities normally should be kept very low (0.01–0.1), otherwise convergence may be delayed unnecessarily. We used the grid search algorithm to determine the best values of crossover and mutation rates. The best value of this two hyperparameters for our proposed model is 0.9 and 0.08, respectively. Due to the superlinear growth of convergence time, we set the population size to 10 times the number of dimensions if the number of dimensions is less than 100. Otherwise, the population size is set to 0.5 times the number of dimensions. The dataset used was the real-world tick data of weather condition belonging to the city of Mashhad, Iran with 525,420 rows and 20 columns. In order to perform the evaluation on a larger dataset, a synthetic dataset with 200 columns was constructed by replication of the number of columns of the abovementioned real tick data. Indeed, there exists real large-scale tick data such as TD-2 which was used in [6] and provided by an investment bank with 500,000 rows and 190 columns. However, we could not obtain it.

In order to evaluate the performance of our proposed model, two criteria were considered. The first criterion is the compression ratio, which, as expressed in the preliminaries section, is the ratio of the storage space of the clusters to the storage space of the original tick data table. Smaller value of compression ratio indicates more efficient compression and thus less required storage space. The second criterion is the execution time, which denotes how fast the tick data table has been clustered. Smaller value of execution time indicates more efficient clustering. The simulation was repeated 10 times (max-iteration = 10) and the average of the obtained results has been measured, as indicated in the following figures and charts (except for Figures 1 and 2 which show the results of one-time simulation execution (max-iteration = 1) as well as 10 times execution.

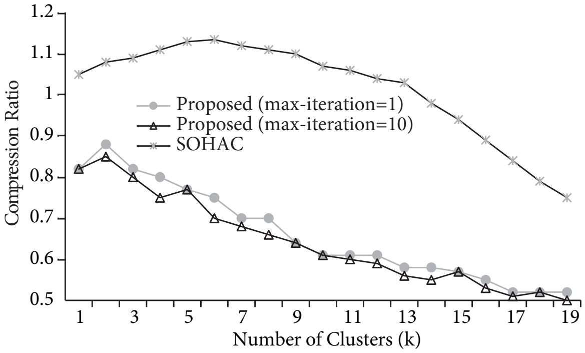

Figure 1. Compression ratio of SOHAC and our proposed clustering method for the small dataset.

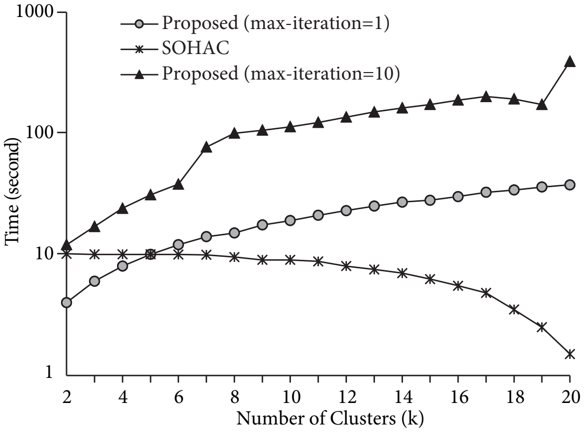

Figure 2. Execution time of SOHAC and our proposed clustering method for the small dataset.

5.2. Simulation results

At first, the clustering results of the small real dataset are demonstrated. Then, the clustering results of the large dataset are presented.

5.3. Clustering and compression the small real dataset

Figure 1 demonstrates the compression ratio with respect to different number of clusters, i.e. k, for SOHAC and our proposed clustering method. As can be observed, for all cluster numbers, our proposed method has obtained smaller values for compression ratio even when max-iteration is equal to 1 (i.e. the experiment is repeated once). As seen in this figure, for smaller values of k (i.e. 2,3, and 4), the compression ratio of SOHAC algorithm exceeds 1, which means that performing the clustering by SOHAC may lead to the increase in the storage size of the tick data. Notice that in each iteration of SOHAC, only all possible 2-combinations of columns or column subgroups are taken into account as potential clusters, while others e.g., 3- combinations of columns or column subgroups are ignored. Thus, sometimes SOHAC cannot cluster or compress data appropriately.

Figure 2 depicts the execution time with respect to k. As can be observed, for smaller values of k (i.e. 2, 3, 4, and 5) the execution time of our proposed model is less than that of SOHAC when max-iteration is equal to 1. In such conditions, the compression ratio of our proposed method is smaller than that of SOHAC. Note that k is usually taken small values because high values of k will incur prohibitive query costs in case of range and point queries that retrieve values for more features [7]. Thus, as mentioned in [7], reasonable values of k are normally in the range of 2 to 5. When max-iteration is equal to 10, the execution time of our proposed model to cluster the small dataset is worse than that of SOHAC. However, as seen in Section 5.4, for large datasets, the execution time of our proposed model is almost better than that of SOHAC even when max-iteration is equal to 10.

5.4. Clustering and compression of the large dataset

Figure 3 depicts the compression ratio with respect to different number of clusters, i.e. k, for the large dataset. As shown in this figure, our proposed model outperforms SOHAC in obtaining better results in terms of smaller compression ratio.

Figure 4 demonstrates the execution time with respect to different number of clusters k. As shown in this figure, our proposed method almost outperforms SOHAC in obtaining smaller execution time, i.e. for $k \in [2, 110]$. The reason is that for k = 2 for example, the time order of SOHAC is $O(m^4 n)$ while the time order of our proposed algorithm is $O(k \times \text{"}\mathit{population\ size}\text{"} \times \text{"}\mathit{no.\ of\ iteration}\text{"} \times n \times m)$. Therefore, if $(k \times \text{"}\mathit{population\ size}\text{"} \times \text{"}\mathit{no.\ of\ iteration}\text{"})$ is much less than m^3, our proposed algorithm is computationally more efficient than SOHAC. When k increases, the execution time of our proposed method increases, and according to Equation 5, the execution time of SOHAC decreases. To be brief, the execution time of our proposed method is worse than that of SOHAC for $k \in [125, 200]$ but the compression ratio of our proposed method is better than that of SOHAC.

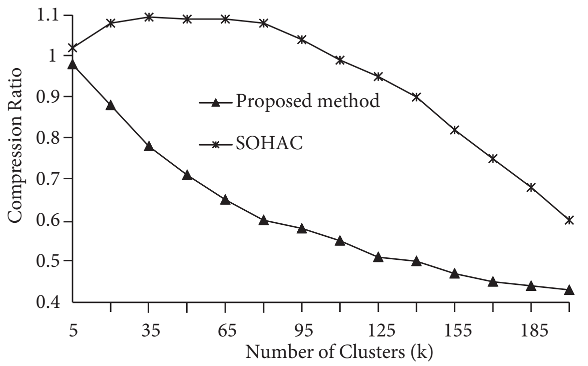

Figure 3. Compression ratio in SOHAC and our proposed clustering method for the large dataset.

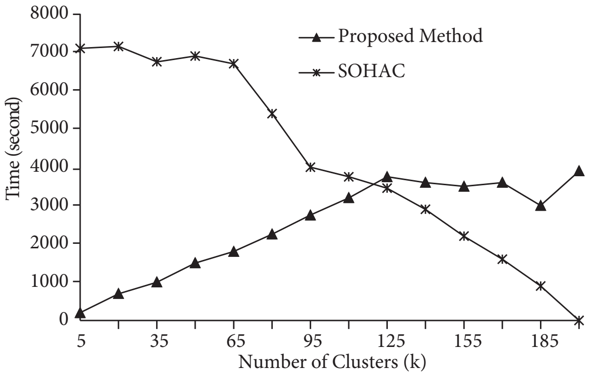

Figure 4. Execution time of SOHAC and our proposed clustering method for the large dataset.

In order to evaluate the compression ratio with respect to different number of dataset columns, we set the number of clusters, i.e. k, to 15 and increased the number of columns from 20 to 200 (by step 10). Figure 5 shows the result of this experiment for both SOHAC and our proposed method. Again, our proposed method achieved smaller compression ratio, compared to SOHAC. Figures 6 and 7 demonstrate the relationship between the number of iterations and the fitness function value for small and large datasets. As it can be seen, for each of the datasets, as the number of iterations increase, the genetic algorithm converges. However, the runtime of genetic algorithm for large dataset is much more than that of small dataset.

6. Conclusion

Tick data has appeared in many applications including financial transactions and climate observations. Since tick data values change rapidly, the size of tick data table grows substantially, leading to storage and query processing challenges. In this article, we presented a new metaheuristic mathematical model for the storage and clustering of tick data. The aim of the proposed method is to partition the tick data table in a way to minimize the required storage space while at the same time efficiently answering the queries. The genetic algorithm has been used to solve the proposed model. In our experiments, we compared our model with SOHAC method on a real-world weather tick dataset. The results demonstrated that our proposed model outperforms SOHAC in

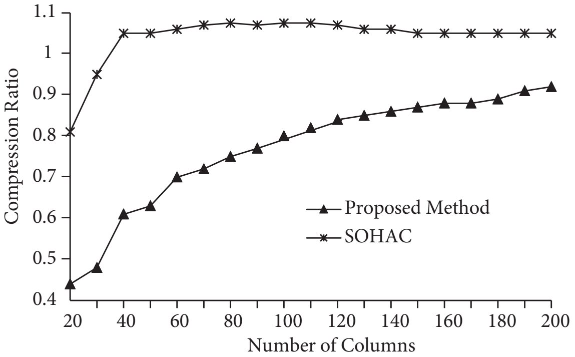

Figure 5. Evolution of compression ratio with different column numbers.

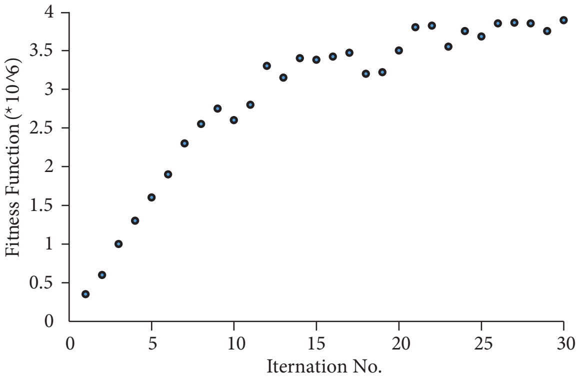

Figure 6. The relationship between the number of iterations and the fitness function value of the GA for the small dataset.

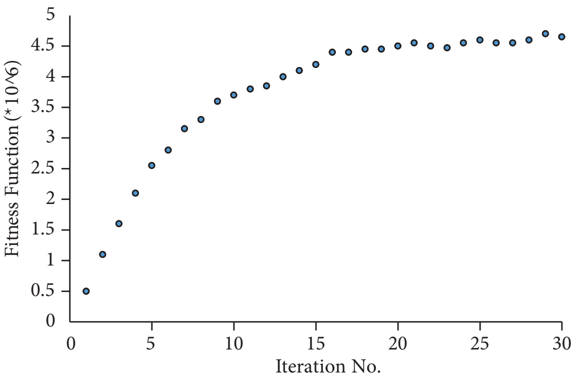

Figure 7. The relationship between the number of iterations and the fitness function value of the GA for the large dataset.

terms of achieving smaller values for compression ratio and performing the clustering in shorter time for small number of clusters.

References

[1] Chen M, Mao S, Liu Y. Big data: a survey. Mobile Networks and Applications 2014; 19(2): 171-209.

[2] Kaur PD. A survey on big data storage strategies. In: Proceesings of International Conference on Green Computing and Internet of Things (ICGCIoT); Noida; 2015. pp. 280-284

[3] Siddiqa A, Hashem IAT, Yaqoob I, Marjani M, Shamshirband et al. A survey of big data management: taxonomy and state-of-the-art. Journal of Network and Computer Applications 2016; (71): 151-166.

[4] Oh KJ, Kim KJ. Analyzing stock market tick data using piecewise nonlinear model. Expert Systems with Applications 2002; 22(3): 249-255.

[5] Salomon D. Data Compression: The Complete Reference. USA: Springer Science and Business Media, 2004.

[6] Nagy GI, Buza K. Sohac: Efficient storage of tick data that supports search and analysis. In: Industrial Conference on Data Mining. New York, NY, USA; 2012. pp. 38-51.

[7] Buza K, Nagy GI. Nanopoulos A. Storage-optimizing clustering algorithms for high-dimensional tick data. Expert Systems with Applications 2014; 41(9): 4148-4157.

[8] Arora S, Chana I. A survey of clustering techniques for big data analysis. In: 5th IEEE International Confluence on the Next Generation Information Technology Summit; 2014. pp. 59-65.

[9] Buza K, Buza A, Kis PB. A distributed genetic algorithm for graph-based clustering. In: Springer Man-Machine Interactions; Berlin, Heidelberg, Germany; 2011. pp. 323-331.

[10] Homaifar A, Qi CX, Lai SH. Constrained optimization via genetic algorithms. Simulation 1994; 62(4): 242-253.

[11] Joines JA, Houck CR. On the use of non-stationary penalty functions to solve nonlinear constrained optimization problems with GA?s. In: Evolutionary Computation, IEEE World Congress on Computational Intelligence; New York, NY, USA; 2015. pp. 579-584.

[12] Dionne G, Duchesne P, Pacurar M. Intraday value at risk (ivar) using tick-by-tick data with application to the toronto stock exchange. Journal of Empirical Finance 2009; 16: 777-792.

[13] Ohnishi T, Mizuno T, Aihara K, Takayasu M, Takayasu H. Statistical properties of the moving average price in dollar–yen exchange rates. Physica A: Statistical Mechanics and its Applications 2004; 344(1): 207-210.

[14] Sazuka N. Analysis of binarized high frequency financial data. The European Physical Journal B-Condensed Matter and Complex Systems 2006; 50(1-2): 129-131.

[15] Pal SH, Patet JN. Time-series data mining: a review. Binary Journal of Data Mining and Networking 2015; 5(1): 01-04.

[16] Ahmad S, Taskaya-Temizel T, Ahmad K. Summarizing time series: learning patterns in volatile series. In: Springer International Conference on Intelligent Data Engineering and Automated Learning; Berlin, Heidelberg; 2004. pp. 523-532.

[17] Nagy GI, Buza K. Partitional clustering of tick data to reduce storage space. In: Proceedings of IEEE 16th International Conference on Intelligent Engineering Systems (INES); Lisbon, Portugal; 2012. pp. 555-560.

[18] Wolpert DH, Macready WG. No free lunch theorems for optimization. IEEE Transactions on Evolutionary Computation 1997; 1(1): 67-82.