Data squashing as preprocessing in association rule mining

Iztok Fister
University of Maribor
Maribor, Slovenia
iztok.fister@um.si

Iztok Fister Jr.
University of Maribor
Maribor, Slovenia
iztok.fister1@um.si

Damijan Novak
University of Maribor
Maribor, Slovenia
damijan.novak@um.si

Domen Verber
University of Maribor
Maribor, Slovenia
domen.verber@um.si

***Abstract*—Data squashing is a well-known preprocessing method in Machine Learning that enables construction of smaller datasets from the original ones and provides approximately the same results of data analysis as the original. The paper proposes a new data squashing method for Association Rule Mining based on the Cosine similarity and Euclidean distance similarity. The method was applied to three datasets from the UCI Machine Learning repository. The results showed that the proposed data squashing method is effective, scalable, and easy to use, and therefore represents a huge potential for use in practice.**

***Index Terms*—association rule mining, data squashing, evolutionary algorithms, machine learning, preprocessing**

1. Introduction

Recently, many large databases emerged that could collect a huge amount of data from various sources, such as social media platforms, IoT sensors, public repositories, cloud storages, logging files, and traditional databases. Such databases demand tremendous time and space performances by operating them on digital computers. Methods for reducing the sizes of the large databases have been developed that require fewer data analysis resources and produce the same results as the original. Data squashing is one of the more important methods in this class.

Data squashing has been defined by DuMouchel et al. [1] as the construction of a smaller dataset providing approximately the same data analysis results as the large dataset. The classical approach in developing these methods tries to preserve the statistical information of data in the reduced database [2]. In general, there are three kinds of data squashing methods: (1) DuMouchel et al. proposed a Taylor series representation for the likelihood function, with which the arbitrary problem is modeled, (2) Madigan et al. [3] approximated a specific likelihood function directly by building the squashed dataset, and (3) Owen [4] developed the empirical likelihood method, capable of direct matching moments of the original variables in the squashed dataset to moments in the original dataset.

The presented study uses Association Rule Mining (ARM) as a testbed to compare different data squashing methods. The ARM is a well-known Machine Learning (ML) method and serves as a tool for discovering the hidden relations between attributes in a transaction database. In this context, a transaction is a recording of some event and associated attributes (e.g., the content of the shopping basket and associated data when a customer presses a buy button in the web application). The ARM generates a set of rules that represents the relation between the attributes in the transactions.

Several modern approaches to ARM are based on evolutionary and swarm intelligence algorithms. Those algorithms allow dealing with numerical and categorical attributes, which is the opposite of most deterministic methods (e.g., Apriori algorithm), which can process only categorical data. Additionally, during the search process, algorithms can be aided with different metrics, which can steer a search process toward the more innovative rules. Readers are invited to read several review papers that highlight recent advances in this research area [5], [6].

In our study, the statistical information of a data set is replaced with a concept of similarity. The idea of the ARM data squashing is to collect similar transactions to the same group (cluster) based on chronological ordering. The chronological ordering means the transactions are collected into groups according to their arrival times. Thus, the first unvisited transaction serves as a centre for creating a similar group, to which the other transactions are attached according to the proposed similarity metrics. Then, the representative squashing vector is computed for each cluster. It bears the characteristics of all the transactions in the group. Also, the weight is attached to this vector that represents the size of a particular cluster, reflecting its significance.

Two similarity metrics were integrated into the proposed squashing method: (1) Cosine similarity and (2) Euclidean distance similarity. The former measures an angle between two vectors, and, consequently, magnitudes of their elements are not taken into regard, while the latter evaluates distance between the same vectors, and, thus, the magnitudes of the elements are important. The proposed method was then applied to three UCI ML datasets [7]. Both variants of the data squashing method have been integrated into a uARMSolver framework [8] running on a Linux server. The results showed a huge potential for use in practice. The proposed method increases the spectrum of the miner's features applicable to the ARM domain.

The main contribution of the paper can be summarized as follows:

- proposing the new preprocessing methods for data squashing based on Cosine and Euclidean distance similarities,
- integrating the method to the uARMSolver framework, thus, widening the usability to a broad spectrum of users, and
- comparing and analyzing the results obtained by two different similarity metrics.

The structure of the remainder of the paper is as follows: Section 2 discusses the background information of the data squashing and association rule mining. In Section 3, the proposed data squashing method is illustrated in detail. The experiments and results are the subjects of Section 4. The paper is concluded with Section 5 that summarizes the performed work and outlines the directions for the future.

2. Background information

The present section explains two main topics relevant to the proposed method: more formal definition of Data squashing and Association rule mining.

2.1. Data squashing

Let us assume a transaction dataset Y is defined as a matrix with N rows representing transactions and K columns denoting different features of a single transaction. Then, the squashed dataset is a matrix X having M rows, where $M \ll N$, and $K+1$ columns. The extra column in X represents weights w_i for $i = 1, \dots, M$. The weights represent the sizes of the data clusters of similar transactions which were created during the data squashing process and where the following criteria need to be satisfied:

$$\sum_{i=1}^{M} w_i = N, \tag{1}$$

subject to $w_i > 0$.

There are many ways to group similar transactions into clusters. From each cluster, a representative transaction is created, which bears the statistical characteristics of all transactions in that cluster. The weights reflect the significance of representative transactions.

2.2. Association Rule Mining

Let assume a set of features $F = \{A_1, \dots, A_C; Q_1, \dots, Q_R\}$, and transaction dataset represented with the matrix D of dimension $N \times K$, are given. A_j for $j = 1, \dots, C$ denotes categorical, and Q_j for $j = 1, \dots, R$ numerical features. Each transaction T is a subset of features $T \subseteq F$. The variable $K = C + R$ indicates the maximum number of features in each row. Each categorical feature A_j is defined by a set of possible attribute values, while each numerical feature Q_j is determined by a closed interval of possible numerical values.

Then, an association rule is defined as an implication:

$$X \Rightarrow Y, \tag{2}$$

where $X \subset F$, $Y \subset F$, and $X \cap Y = \emptyset$. The quality of the association rule is evaluated using several metrics [9]. Three of them were used in the research.

The support metric represents the number of transactions that includes both the antecedent and consequent parts of the rule, divided by the number of all transactions:

$$supp(X \Rightarrow Y) = \frac{n(X \cup Y)}{N}. \tag{3}$$

Confidence is the conditional probability of occurrence of the consequent given the antecedent. It is calculated as ratio between the number of transactions that includes both the antecedent and consequent parts rule and the number of the transactions with the antecedent

$$conf(X \Rightarrow Y) = \frac{n(X \cup Y)}{n(X)}. \tag{4}$$

The inclusion refers to the ratio of the size of the antecedent and the size of the consequent to the total number of features:

$$incl(X \Rightarrow Y) = \frac{1}{K}\Big[ante(X \Rightarrow Y) + cons(X \Rightarrow Y)\Big]. \tag{5}$$

The parameter N in equation (3) represents the number of transactions in the dataset D, and $n(.)$ is the number of elements in a set. The functions $ante(.)$ and $cons(.)$ count the number of attributes in the antecedent and consequent, respectively. The variable K is the total number of features in the transaction database.

Additionally, C_{min} denotes minimum confidence and S_{min} minimum support, determining that only those association rules with confidence and support higher than $conf(X \Rightarrow Y) \geq C_{min}$ and $supp(X \Rightarrow Y) \geq S_{min}$ are taken into consideration, respectively.

3. ARM data squashing

The ARM data squashing aims to reduce the original transaction dataset's size so that it keeps the same characteristics as the original one. Consequently, the results of some data analysis made on the reduced dataset are approximately the same as those made on the original. In squashing the datasets, two issues need to be considered:

- how to group the similar transactions into the clusters, and
- how to construct the transaction that represents the characteristics of all transactions in the cluster as reliably as possible.

In the research, two measurements of the similarity between vectors $\mathbf{x}$ and $\mathbf{y}$ that form the same cluster were used:

- the Cosine similarity, calculated according to Cauchy-Schwartz inequality [10], and
- the Euclidean distance similarity [11].

In the end, the proper representative transaction that bears the characteristics of the specific cluster are constructed. For that, the average value of each element of the cluster transactions is calculated. The bigger the size of the cluster, the more expressive power it has.

In the remainder of the section, both similarity metrics are explained in detail. Then, the design of an ARM squashing algorithm is discussed. Finally, the implementation of the algorithm is presented.

3.1. Similarity metrics

The Cosine similarity metric is expressed as follows:

$$sim_{\cos}(\mathbf{x},\mathbf{y}) = \frac{|\mathbf{x}\cdot\mathbf{y}|}{||\mathbf{x}||\cdot||\mathbf{y}||}, \tag{6}$$

where $|\mathbf{x}\cdot\mathbf{y}|$ is a scalar product of two vectors, and $||\mathbf{x}||$ and $||\mathbf{y}||$ denote the norms of both vectors.

The Euclidean distance similarity is defined as:

$$sim_{\mathrm{Eucl}}(\mathbf{x},\mathbf{y}) = \sqrt{\sum_{i=1}^{N}\frac{(x_i-y_i)^2}{(x_i^{\max}-x_i^{\min})^2}}, \tag{7}$$

where $\mathbf{x}=\{x_i\}$ and $\mathbf{y}=\{y_i\}$ represent items in the original transaction dataset, $\mathbf{x}^{\max}=\max_{i=1,\ldots,N}\{x_i\}$ and $\mathbf{x}^{\min}=\min_{i=1,\ldots,N}\{x_i\}$ are the maximum and minimum values of the vector's elements, and N the number of transactions. The normalization factor is used to keep the value of the similarity between zero and one.

3.2. Design of the ARM data squashing algorithm

The ARM data squashing algorithm was integrated into a universal framework [8] called the uARMSolver. The framework was developed in the C++ programming language. Its main advantages are speed, modular design, and open-source availability. This framework provides for all three steps in the ARM: preprocessing, optimization, and visualization. The squashing algorithm supplements the existing preprocessing methods of the tool.

Three issues must be considered when integrating the ARM data squashing into the uARMSolver:

- examining the size of the specific clusters,
- modifying the fitness function accordingly, and
- determining the representative transaction in a cluster.

The first issue is considered by including the additional feature in the transaction dataset that identifies the weight of the cluster. The weight of the cluster is later incorporated into the fitness function evaluation as follows:

$$f(X\Rightarrow Y) = \frac{w_i}{\alpha+\beta+\gamma}\cdot[\alpha\cdot supp(X\Rightarrow Y)+ \\ \beta\cdot conf(X\Rightarrow Y)+\gamma\cdot incl(X\Rightarrow Y)], \tag{8}$$

where w_i presents the weight of the representative rule in the squashed transaction dataset, α, β, and γ denote weights of definite metrics in the equation; $supp(X\Rightarrow Y)$, $conf(X\Rightarrow Y)$, and the term $incl(X\Rightarrow Y)$ represents the support, the confidence, and the inclusion of the observed association rule, respectively.

The elements within the transaction are utilized in the proposed algorithm depending on the feature type. When a categorical feature is observed, the frequency of appearances of an observed attribute is counted, and then the most frequent attribute is taken into the representative transaction. Consequently, the averages of the elements lying in the same position in the cluster members are calculated to obtain the representative elements of numerical types.

3.3. Implementation of the ARM data squashing algorithm

The main principle of the proposed ARM data squashing algorithm is to preserve the original ordering of the transactions. Therefore, the transactions are processed in order from the start of the dataset to the end. The already processed values are marked as visited. In each step, firstly, the first unvisited transaction is selected. That represents the first transaction in the new cluster. After that, other unvisited transactions that satisfy the relation:

$$sim_{\cos|\mathrm{Eucl}}(\mathbf{x},\mathbf{y}) \le Thresh \tag{9}$$

are added. The variable $Thresh$ denotes the minimum similarity value allowing the specific transaction to be added to the cluster. The process is repeated until all unvisited transactions are checked. Next, a representative transaction of the cluster is calculated. Finally, the representative transaction is added to the squashed dataset. The procedure is repeated with the next cluster and is terminated when the last unvisited transaction is processed.

The pseudo-code of the ARM data squashing algorithm is illustrated in Algorithm 1, from which it can be seen that

Algorithm 1 The proposed data squashing algorithm.

```
1: $sqTran = \emptyset$
2: for $i = 1$ to $nTran$ step 1 do
3:     $visited[i] = $ false
4: for $i = 1$ to $nTran$ step 1 do
5:     if IsVisited($i$) then continue
6:     $setTran = \emptyset$
7:     AddTran($setTran, Tran[i]$)
8:     $visited[i] = $ true
9:     for $j = i+1$ to $nTran$ step 1 do
10:        if IsVisited($j$) then continue
11:        $sim$=Similarity($Tran[i], Tran[j]$)
12:        if $sim < Thresh$ then
13:            AddTran($setTran, Tran[j]$)
14:            $visited[j] = $ true
15:    MakeSqVector($sqVector, setTran$)
16:    AddTran($sqTran, sqVector$)
```

the proposed algorithm consists of two **for** loops, where the outer one selects the first unvisited transaction in the original ARM dataset representing the first element of the new cluster (function ADDTRAN at line 7). The inner **for** loop add all the other unvisited transactions satisfying the required relation to the cluster (function ADDTRAN at line 13). Then, the representative vector is calculated finally (function MAKESQVECTOR at line 15) that is added into the squashed dataset (function ADDTRAN at line 16).

The outcome of the algorithm depends on the parameter $Thresh$. The lower the variable, the more clusters are created, and vice versa and that determines the final size of the squashed dataset.

4. Experiments and results

Our experimental work aimed to show that the proposed data squashing methods for ARM can achieve good results according to space and time limitations for its application in practice. For that, the proposed data squashing methods were integrated into uARMSolver [8]. The experiments were conducted in two phases: (1) preprocessing of the data and (2) mining the association rules with the Differential Evolution (DE) algorithm. Let us emphasize that the implementation of this algorithm is a part of the framework, and, therefore, a detailed description of this algorithm is out of the scope of the paper.

The parameters of the DE algorithm were set as follows: The population size $Np = 100$ was determined during the experimental work. The other two DE algorithm parameters were set as proposed by the DE community (i.e., the scale factor $F = 0.9$, and the crossover rate $CR = 0.5$). As a termination condition, the convergence window $w = 100$ was used, which terminates the algorithm after 100 generations in which no progress was detected in improving the fitness function value.

The test-bed for evaluating the results obtained by the proposed method is presented by three UCI ML datasets [7] as illustrated in Table 1. The table shows the characteristics

TABLE 1. CHARACTERISTICS OF THE UCI ML DATASETS.

Nr.	Dataset	Nr.transactions	Nr.features	Type
1	Abalone	4,177	9	Mixed
2	Mushroom	8,125	22	Categorical
3	Adult	32,561	14	Mixed

of the used datasets: the name of the dataset, the number of transactions, the number of features, and their corresponding types. As seen from the table, two datasets contain mixed (i.e., categorical and numerical) attribute types and the third only the categorical ones. At first, the datasets were preprocessed and then processed by the uARMSolver

Several experiments were conducted in the study. The results were analysed according to: (1) the quality and (2) the time complexity. The remainder of the section presents the detailed results. The section concludes with a critical analysis of the results concerning the similarity metrics.

4.1. Detailed results regarding the quality

The results of the data squashing applied to the Abalone dataset using the Cosine similarity metric are presented in Table 2, which consists of columns giving the size of the

TABLE 2. SQUASHING OF ABALONE DATASETS USING COSINE SIMILARITY.

Thresh	SquashDB	SquashRate	bestFitness	Diff. [%]
0.75	2	99.9521	0.962963	0.0083
0.8	2	99.9521	0.962963	0.0083
0.85	2	99.9521	0.962963	0.0083
0.9	2	99.9521	0.962963	0.0083
0.95	3	99.9282	0.962963	0.0083
0.99	10	99.7606	0.962963	0.0083
0.999	57	98.6354	0.957115	-0.5990
0.9999	487	88.3409	0.962279	-0.0627
0.99999	1123	73.1147	0.962666	-0.0225
0.999999	4,164	0.3112	0.962883	0.0000
1.000000	4,177	0.0000	0.962883	0.0000

squashed dataset, the squashing rate, the best fitness obtained by the uARMSolver, and the difference with the original non-squashed dataset at $Thresh = 1.0$. The table presents the results for different settings of the $Thresh$ parameter.

As can be seen from the table, the influence of the $Thresh$ parameter is crucial. However, the correlation between the $Thresh$ parameter and the size of the squashed database is not linear. The difference between the approximated value of the fitness function value obtained on the squashed dataset and the original one obtained by the uARMSolver is less than 1 $\%$ by all the observed instances.

The results of the data squashing on the same database but using the Euclidean distance similarity are illustrated in Table 3, from which it can be seen that some linear correlation

TABLE 3. SQUASHING OF THE ABALONE DATASETS USING EUCLIDEAN DISTANCE.

Thresh	SquashDB	SquashRate	bestFitness	Diff. [%]
0.50	11	99.7367	0.962963	0.0083
0.55	18	99.5691	0.962963	0.0083
0.60	23	99.4494	0.962963	0.0083
0.65	38	99.0903	0.954191	-0.9027
0.70	55	98.6833	0.956902	-0.6211
0.75	76	98.1805	0.958577	-0.4472
0.80	121	97.1032	0.960208	-0.2778
0.85	246	94.1106	0.961608	-0.1324
0.90	627	84.9892	0.962431	-0.0469
0.95	2,262	45.8463	0.962816	-0.0069
1.00	4,177	0.0000	0.962883	0.0000

exists between the setting of the $Thresh$ parameter and the size of the squashed database. On the other hand, the approximate best fitness function values obtained by varying the $Thresh$ parameter are distinguished by their original value at the instance $Thresh = 1.0$ for less than 1 $\%$.

Let us also mention that some negative values of the differences can be observed in Tables 2-3. These negative values show that the ARM mining algorithm didn't achieve the original bestFitness value in some instances of $Thresh$ values. For example, the instance $Thresh = 0.999$ in Table 2 has performed 0.6 $\%$ less quality solution according to bestFitness when squashing database for more than 98 $\%$. This means that, in place of mining the original transaction database with 4,177 transactions, the ARM mining algorithm conducted on the squashed database with 57 transaction sacrifices the bestFitness value for only $0.6\ \%$.

The detailed results regarding the quality obtained by mining the Mushroom and Adult databases are presented graphically in Figures 1 and 2, respectively. The x-axis denotes the number of transactions and is logarithmic to emphasise the small changes at the beginning.

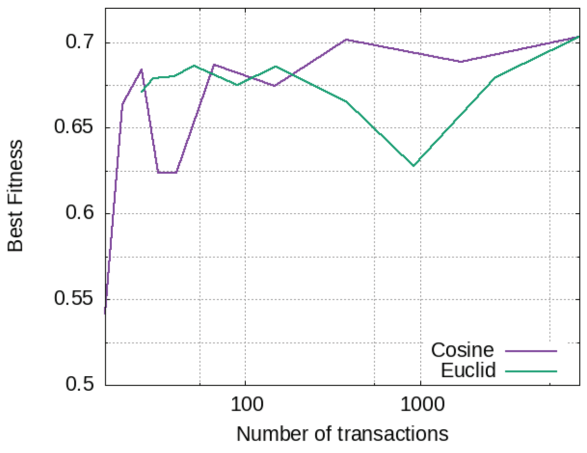

Figure 1. Squashing of the Mushroom dataset.

As seen from Figure 1, the Cosine similarity metric is more sensible on the lower values of the $Thresh$ parameter. On the other hand, this similarity directs the algorithm to faster convergence.

This fact is emphasised even more in Figure 2, where the best fitness function values deviate a lot around the best value by the lower settings of this parameter before convergence is achieved. The convergence is achieved immediately when

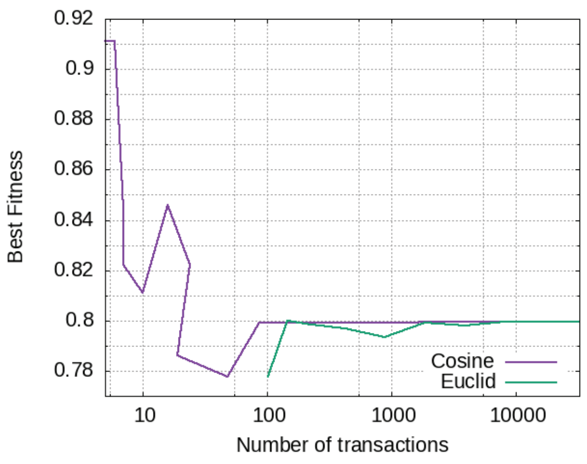

Figure 2. Squashing of the Adult dataset.

the value of the $Thresh$ parameter is changed from 0.99 to 1.00. This small change of parameter results in an enormous change in the number of transactions in the squashed dataset. In our case, from the 87 (or $0.46\ \%$) to 32,562 (or $100.00\ \%$).

4.2. Detailed results regarding the time

Data squashing methods influence the execution time of data processing greatly. The smaller the squashed dataset, the faster it will be processing. The execution time-related results of squashing the Adult database using the Cosine similarity are illustrated in Table 4, that is divided into more

TABLE 4. SQUASHING OF THE ADULT DATASETS USING COSINE SIMILARITY.

Thresh	SquashDB	Time	Total	Diff. [%]
0.70	5	0.667	1.956	99.94
0.75	6	1.104	2.685	99.92
0.80	7	0.297	1.652	99.95
0.85	7	0.976	2.342	99.94
0.90	10	0.549	2.394	99.93
0.95	16	1.569	5.001	99.85
0.96	24	0.333	2.389	99.93
0.97	19	0.371	2.213	99.93
0.98	48	0.562	4.391	99.87
0.99	87	1.941	10.735	99.67
1.00	32,562	600.207	3275.61	0.00

columns that represent: the size of the squashed dataset, the time needed for finding the best solution, the total time required for satisfying the termination condition, and the difference between the total time of the uARMSolver and specific execution time of solving the problem as determined by the parameter $Thresh$. Ten measurements are obtained by varying the threshold parameter $Thresh \in [0.70, 0.99]$, while the last (at $Thresh = 1.0$) denotes the results of the uARMSolver on the original dataset.

From Table 4, it can be seen that the execution time is decreased by data squashing by more than $99\ \%$. However, because the dataset size is reduced significantly, it could be speculated that much information is lost (Figure 2).

The situation is changed significantly when the results of the same squashing are observed according to the Euclidean distance similarity (Table 5). The threshold is varied in

TABLE 5. SQUASHING OF ADULT DATASETS USING EUCLIDEAN DISTANCE.

Thresh	SquashDB	Time	Total	Diff. [%]
0.50	102	1.662	13.526	99.59
0.55	146	2.294	20.224	99.38
0.60	226	2.102	34.187	98.96
0.65	425	7.865	53.513	99.59
0.70	880	29.061	113.479	96.54
0.75	1,862	37.694	196.518	94.00
0.80	3,919	55.118	405.636	87.62
0.85	8,201	139.687	837.709	74.43
0.90	15,311	370.330	1617.290	50.63
0.95	26,279	307.887	2442.270	25.44
1.00	32,562	600.207	3275.610	0.00

the interval $Thresh \in [0.5, 0.95]$ in steps of 0.05, and ten measurements were obtained.

It can be seen from Table 4, that both the size of the dataset and the execution time correlates with the parameter $Thresh$. Interestingly, the quality of the results (Figure 2) shows that these do not change significantly when the size of the squashed database is higher than 100 transactions, i.e., when $Thresh > 0.5$.

4.3. Discussion

The proposed data squashing method based on two similarity metrics has shown their potential for application to ARM problems in practice. Nevertheless, there are some pros and cons of the used method, which are summarised here:

1) The small changes of the threshold parameter $Thresh$ by the Cosine similarity can generate the huge changes in the size of the squashed dataset.
2) On the other hand, the Euclidean distance similarity measures the distances between elements of the observed vectors in the problem search space. Consequently, this metric is more linear, scalable, and therefore easier for use in practice.
3) However, the results of different metrics cannot be distinguished according to a t-test 2-paired parametric statistical test (Table 6). Although the threshold value affects the size of the squashed dataset for both metrics differently, the results were not significantly different.

TABLE 6. THE RESULTS OF THE t-TESTS.

Database	t-value	p-value	$p < 0.01$
Abalone	1.48681	0.152661	No
Mushroom	-0.91525	0.370967	No
Adult	2.42356	0.024976	No

4) The smaller-squashed datasets were obtained using the Cosine similarity. This is fine for the datasets with numerical attributes but can break down with the dataset containing mixed attributes like in Figure 1.

Cosine similarity is generally a good choice for high dimension and categorical data, while the Euclidean-based similarity measure is a good choice for numerical datasets. In addition, the Euclidean distance is not a good choice for a dataset that contains categorical attributes.

5. Conclusion

The paper introduces a new data squashing method for ARM based on Cosine and Euclidean distance similarity metrics. The method is devoted primarily to reducing the sizes of transaction datasets used for mining the association rules.

The proposed method was integrated into the uARM-Solver framework. Three datasets from the UCI ML repository served as a test-bed for testing the method. Despite its simplicity, the proposed metrics reduced the original datasets significantly without losing the information quality.

In the future work, the automatic setting of the threshold parameter, which has a crucial impact on the results, will be studied. A deeper insight into the parameter setting could be obtained by analysing the other, especially large-sized, UCI ML datasets.

References

[1] W. DuMouchel, C. Volinsky, T. Johnson, C. Cortes, and D. Pregibon, "Squashing flat files flatter," in *KDD '99*, 1999.

[2] W. DuMouchel, *Data Squashing: Constructing Summary Data Sets*. Boston, MA: Springer US, 2002, pp. 579–591. [Online]. Available: https://doi.org/10.1007/978-1-4615-0005-6_16

[3] D. Madigan, N. Raghavan, I. Raghavan, W. Dumouchel, M. Nason, C. Posse, and G. Ridgeway, "Likelihood-based data squashing: A modeling approach to instance construction." *Data Mining and Knowledge Discovery*, vol. 6, pp. 173–190, 1999.

[4] A. Owen, "Data squashing by empirical likelihood," *Data Mining and Knowledge Discovery*, vol. 7, pp. 101–113, 1999.

[5] I. Fister Jr and I. Fister, "A brief overview of swarm intelligence-based algorithms for numerical association rule mining," *Applied Optimization and Swarm Intelligence*, pp. 47–59, 2021.

[6] E. Varol Altay and B. Alatas, "Performance analysis of multi-objective artificial intelligence optimization algorithms in numerical association rule mining," *Journal of Ambient Intelligence and Humanized Computing*, vol. 11, no. 8, pp. 3449–3469, 2020.

[7] D. Dua and C. Graff, "UCI machine learning repository," 2017. [Online]. Available: http://archive.ics.uci.edu/ml

[8] I. Fister and I. F. Jr., "uarmsolver: A framework for association rule mining," *CoRR*, vol. abs/2010.10884, 2020. [Online]. Available: https://arxiv.org/abs/2010.10884

[9] R. Agrawal and R. Srikant, "Fast algorithms for mining association rules in large databases." Morgan Kaufmann Publishers Inc., 1994, pp. 487–499. [Online]. Available: http://dl.acm.org/citation.cfm?id=645920.672836

[10] S. Lipschutz, *Schaum's Outline of Theory and Problems of Linear Algebra*. New York: McGraw-Hill, 2009.

[11] L. Liberti and C. Lavor, *Euclidean Distance Geometry: An Introduction*. Berlin: Springer Nature, 2017.