ISSN 1330-3651 (Print), ISSN 1848-6339 (Online)

https://doi.org/10.17559/TV-20171109130510
Original scientific paper

Improved Visualization of Frequent Itemset Relationships Using the Minimal Spanning Tree Algorithm

Mihaela VRANIĆ, Damir PINTAR, Frano ŠKOPLJANAC-MAČINA

Abstract: Descriptive data mining techniques offer a way of extracting useful information out of large datasets and presenting it in an interpretable fashion to be used as a basis for future decisions. Since users interpret information most easily through visual means, techniques which produce concise, visually attractive results are usually preferred. We define a method, which converts transactional data into tree-like data structures, which depict important relationships between items contained in this data. The new approach we propose is offering a way to mitigate the loss of information present in previously developed algorithms, which use mined frequent itemsets and construct tree structures. We transfer the problem to the domain of graph theory and through minimal spanning tree construction achieve more informative visualizations. We highlight the new approach with comparison to previous ones by applying it on a real-life datasets – one connected to market basket data and the other from the educational domain.

Keywords: association rules; data mining; dendrograms; frequent itemsets; minimal spanning tree; transactional data; visual representation

1 INTRODUCTION

Modern businesses commonly generate huge amounts of data on daily basis. A significant portion of this data is transactional data, a term denoting data, which describes a specific business event, which happened at a certain point in time (a typical example of transactional data may be point-of-sale records in a fast-moving consumer goods chain). Analysing this data can offer beneficial insights which can help business experts in coming up with future business decisions. One of the most well-known methods for this purpose is market basket analysis based on association rules presented in [1]. This method takes transactional data as input and outputs clear and easy interpretable rules in "if-then" form (e.g. if customer buys products A and B, we can expect with some confidence she will also buy C and D). However, it is common that this method results in a huge number of rules with high level of redundancy and no easy way to discern interesting findings from irrelevant conclusions. A concise, readily interpretable visualization is often preferable by business analysts and clients.

There are many papers devoted to the visualization of transactional data. One of the offered solutions can be found in [2] presenting developed strategies for visual representation of so-called" closed frequent itemsets" which are mined from a transactional dataset given the chosen support parameter. Since concise visualization inevitably contains a certain loss of information, numerical measure which can help the analyst in estimating how much information is contained in the visual structure compared to the complete collection of closed itemsets has been devised. In this paper we go one step further, providing an additional mechanism to showcase closed frequent itemsets which loosens up certain restrictions imposed by original algorithms. To achieve this, we orient ourselves toward transferring the problem to the domain of graph theory, leveraging algorithms for the creation of minimal spanning trees and fitting all closed frequent itemsets into graph structure with the main focus being that they are easily presented in a two-dimensional plane. For implementation and validation of our approach, we have decided to move away from the tool described in [2] and reimplement everything from scratch using R programming language. Our new implementation integrates previously developed strategies with the algorithm presented in this paper. This new implementation enables a higher level of flexibility and reproducibility and is currently openly available at [3]. In addition, easier comparison between developed approaches is enabled. Methods were tested on various datasets, which are representative examples of transactional data. Besides the most common approach of analysing product sales in commercial domain, another interesting area of interest might be the educational domain where the educational data may be reshaped into transactional form and then analysed to offer useful insights in the knowledge acquisition process, potentially useful for integration into e-learning solutions.

The paper is structured as follows. Related work and known visualization methods regarding transactional data are presented in Section 2. In Section 3 methodology is presented together with the most important terms and definitions from graph theory. More details on transferring the problem from transactional domain to graph theory is also provided here. In Section 4 usage of minimal spanning tree to showcase relationships between closed frequent itemsets is examined. Section 5 gives overview of implementation while section 6 brings results evaluation. Newly developed algorithm was examined and presented on three datasets: a small, "toy" dataset, a typical transactional dataset representing purchases in a computer shop and a dataset gained from educational domain and transformed to appropriate form. Finally, in section 7 we present our conclusions and future work.

For the purposes of evaluating contribution of individual authors to this paper, should the need for such evaluation arise, the first two listed authors have contributed equally to the majority of research work which this paper pertains to, and should both be considered as having the first author status.

2 RELATED WORK

Association rule mining, one of the staples of descriptive data mining analysis, was first introduced in [1]. It enables a transformation of a transactional dataset into a set of rules in "if-then" form, which reveal interesting

relations between variables described in the dataset. A most common example is market basket analysis, in which each transaction represents a list of items purchased by a customer on a checkout, and association rules may reveal potentially useful statements such as "if customer bought products A and B, he will almost certainly buy products C and D".

One of the drawbacks of the initial approach was reliance on a resource-intensive algorithm to produce so-called "frequent itemsets", which are intermediate results used by the algorithm in the process of discovering rules. Subsequent research yielded more efficient algorithms for this purpose, with some of the most recent approaches being described in [4] and [5].

Another drawback stems from the fact that the method results in a potentially large set of rules which may contain lots of redundancy and no clear guidance on how to filter "interesting" rules from the uninteresting ones. There are plenty of approaches to tackle this issue. For example, [6] and [7] use the approach of reducing the number of analysed items by grouping them at a higher hierarchical level, according to some existing taxonomy. [8] on the other hand provides a survey on the measures of interestingness together with suggestions how to use them in data mining research.

Since people tend to more easily process visual information than textual, one may wonder whether a list of rules given in text is really the optimal way to visually present the rules to the analyst. We argue that the following statement by [9] still holds: one of the biggest challenges of modern data mining is the issue of presentation and visualization of the results in order to facilitate the process of finding new, previously unknown connections and relationships. One of the papers which employ this approach is [10], using a so-called "structured association map" to visualize transactional data.

In our previous work we have tackled this issue directly – delivering information commonly provided by association rules in a visual, concise, easily interpretable fashion, while achieving enough flexibility to allow the analyst to influence both the parameters of the analysis and the properties of the final results, with enough insight in various side-effects of this approach, such as estimated information loss due to conciseness. Our initial research of transaction element grouping into dendrogram structures [11] was followed by developing measures for hierarchical clustering of transactional data [12]. In [2] we set a goal to transform a transactional dataset into tree-like structures which would represent this dataset in its entirety and as such provide a simplified yet informative high-level view on interesting relationships hidden within – not as a set of "if-then" rules, but as connected nodes in a tree. In that paper we have described two algorithms for generation of these dendrogram structures: a so-called "top-down" strategy, which progressively breaks down relationships between larger itemsets and their components, and a "bottom-up" strategy, which starts from singular items and builds a tree based on their relationships with increasingly larger itemsets. We have developed a custom analysis tool, which implemented these strategies and allowed interactive exploration of the results.

This paper is a direct continuation of research presented in that article. Here we address certain shortcomings and compromises which improve the approach used in [2] by offering new methods oriented mostly towards smaller datasets and further reduction of information loss. This way we will be expanding the toolbox offered to the analyst and allow for a more flexible and customizable approach to transactional data analysis.

3 METHODOLOGY

3.1 Research Goals

As mentioned in the previous section, the most important research goals described in this paper deal with improving strategies from [2], based on suggestions and identified shortcomings gathered from direct feedback made available to us by the analysts who were using our algorithms and tools in various production environments. Two of the most common suggestions were:

- improving the "drill-down" functionality of the algorithm, making it easier for the analyst to investigate further just a subset of frequent itemsets, especially concerning itemsets and relationships which were not included in the visualization initially;
- further reduction of information loss when dealing with smaller datasets, where the more "cluttered" visualization is an acceptable compromise considering the added information provided by the visualization.

To achieve these goals, we have suggested two following approaches:

- parameterized dendrogram construction where the analyst can manually choose the itemset as a root, without the need for previous dataset pre-processing or restarting the frequent itemset identification;
- introducing the minimal spanning tree algorithm which allows a more complete relationship representation with reduced information loss (but with compromised conciseness, which is acceptable in datasets with a smaller amount of attributes).

Necessary graph theory definitions for concepts used in our approach as well as a detailed description of our methodology to achieve the described research goals are presented in the next subsections.

3.2 Important Terms and Definitions from Graph Theory

In order to use the principles and algorithms already developed within graph theory, it is necessary to provide some basic definitions. The following are important definitions that are the sum of definitions and statements based on literature [13-16].

Definition 1: A **simple graph** G consists of a non-empty finite set $V(G)$, whose elements are called the **vertices** (also called nodes or points) of the graph G and the finite set of $E(G)$ which are 2-element subsets of V called the **edges** (also called arcs or lines) (i.e., an edge is associated with two vertices, and the association takes the form of the unordered pair of the vertices).

A simple graph, by definition, excludes the possibility of joining two nodes with multiple edges and the possibility of having loops - bridges connecting one node

to itself. In a simple graph with n vertices, the degree of every vertex is at most $n - 1$.

Definition 2: For **edge** $e = \{v, w\}$ we say that it connects vertices v and w and, without possibility of confusion, shorter we write vw. We say that v and w are **adjacent** vertices.

Definition 3: For given disjoined graphs $G_1 = (V(G_1), E(G_1))$ and $G_2 = (V(G_2), E(G_2))$ their **union** $G_1 \cup G_2$ is given as $G_1 \cup G_2 = (V(G_1) \cup V(G_2), E(G_1) \cup E(G_2))$.

Definition 4: The graph is **connected** if it cannot be displayed as a union of two disjunct graphs. Otherwise, we say the graph is **unconnected**. Each **unconnected** graph can be displayed as a union of **connected** graphs.

Simply put, when the graph can be shown as a union of some two disjunct graphs, then it certainly does not contain any edges connecting two vertices from different graphs that make up the mentioned union. An example of unconnected graph is given in Fig. 1. Here graph G_3 can be seen as a union of two graphs G_1 and G_2 that are each connected graphs.

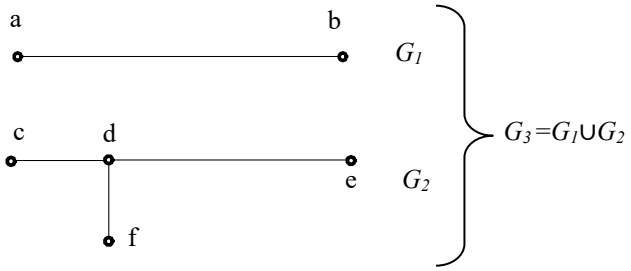

Figure 1 Example of union of two connected graphs, being itself an unconnected graph

Definition 5: Let us denote the vertices of the given graph G with $V = \{1, 2, \ldots, n\}$, then we define the **adjacency matrix** $A = [a_{ij}]$ as a $n \times n$ matrix whose element a_{ij} is equal to the number of edges connecting the node i and the node j.

For a simple graph, the adjacency matrix is a symmetric matrix whose elements have a value of 0 or 1.

Definition 6: The **walk** in the given graph G is a finite sequence of edges v_0v_1, v_1v_2, …, $v_{m-1}v_m$. Frequently it is denoted as: $v_0 \rightarrow v_1 \rightarrow \ldots \rightarrow v_m$. Each subsequent consecutive edge is either adjacent or equal. The vertex v_0 is called the starting node or the source, while the node v_m is called the end node of the walk. The number of walking edges is called walking length m.

A walk is an alternating sequence of vertices and connecting edges.

Less formally, a walk is any route through a graph from vertex to vertex along edges. A walk can end on the same vertex on which it began or on a different vertex. A walk can travel over any edge and any vertex any number of times.

Definition 7: A **trail** is a walk that does not pass over the same edge twice. A trail might visit the same vertex twice, but only if it comes and goes from a different edge each time. A **path** is a walk that does not include any vertex twice, except that its first vertex might be the same as its last. A **cycle** is a path that begins and ends on the same vertex.

Definition 8: A **forest** is an undirected acyclic graph. Connected forest is called a **tree**. In other words, all of connected components of a forest are trees.

Definition 9: A **weighted graph** is a graph in which a number (the weight) $w(e)$ is assigned to each edge. Weight w is a function w: $E(G) \rightarrow \mathbf{R}$ and it might represent for example costs, lengths or capacities, depending on the problem at hand.

Definition 10: Graph $G' = (V', E')$ with $V' \subseteq V$ and $E' \subseteq E$ is called **subgraph** of G.

Definition 11: Spanning tree T of a graph G is a subgraph of G that is a tree and includes all of the vertices of G, that is $V' = V$.

Every connected graph has a spanning tree that can be built on it. Spanning tree from a related graph can be formed in several ways. One of the tree-forming algorithms can be realized in such a way that all nodes of the graph are separate components which are then connected to the tree by gradually including certain edges. The pseudo code of this algorithm is given below:

Algorithm for adding edges:

- let $S = \emptyset$
- **while** graph G (V, S) is unconnected
    - **find** edge e which connects nodes in different components
    - **add** e in S
- **return** (V, S)

The algorithm that works on this principle is given in [15]. A slightly different principle of building a tree over graph G is given in [16]. The algorithm that is displayed therein can be written in pseudocode as follows:

Algorithm for adding neighbour nodes:

- **choose any** node $v \in V$, add v to V', $S = \emptyset$
- **until** $V' \neq V$
    - **find** edge e which connects node { i } $\in V'$ and node { j } $\in (V - V')$
        - **add** e to S
        - **add** j to V'
- **return** (V, S)

One graph can generally have many spanning trees.

Definition 12: Minimal Spanning Tree (**MST**) of a weighted graph is a tree built on it where the tree weight, that is the sum of included edge weights, is minimal or equal to the weight of every other spanning tree.

Spanning trees are interesting for the purpose of this paper because they can always be displayed planary without intersecting edges. If we have a problem that can be modelled using graphs and if we are interested in just the minimal spanning tree builtupon this graph, we can solve this problem in this domain using algorithms developed within the graph theory. In order for them to be adequately presented, it is necessary to introduce two most common algorithms developed for the formation of MST. One is called Prim's algorithm (presented in [17]) and the other is the Kruskal's algorithm (presented in [18]). These algorithms differ in their approach to MST's construction. Prim algorithm is based on the *adding neighbour node algorithm* as described above, but with the requirement of

adding an edge with minimum weight. The Kruskal algorithm works on the principle of the above-mentioned *algorithm of adding edges*, i.e. assuming that all nodes are fragments that merge into larger fragments by adding at each step those bridges that have minimal weight and do not create a cycle. Consequently, in a series of steps, we have a forest whose parts are connected at each step to eventually form the minimal spanning tree.

The algorithm that is more suitable for application in our situation is Kruskal's algorithm, so its pseudo code is given in Fig. 2. Prim's algorithm would not be appropriate because it could not result with the structure of separate tree fragments – which is the structure we often expect to get in analysis on datasets interesting to us.

```
Input:  G=(V(G), E(G)), graph with defined sets of nodes
        and edges, number of edges in stored within
        variable edge number
        w, every edge has assigned a certain weight
Output: G_MST=(V(G), E'(G)), minimal spanning tree over
        G
E'=∅;
edge_number =0;
order edges ascending according to their weights;
for every edge e∈E starting with smallest weight towards
greater {
    if e is not forming a cycle with edges within E'
        then {
            edge_number ++
            add e to E';
        }
    If edge_number =node_number -1 then break the loop;
}
returnG_MST=(V, E');
```

Figure 2 Kruskal's algorithm for MST formation

3.3 Transferring the Problem from Transactional to Graph Domain

In order to meet the required goals we need to translate the problem from initial domain (the domain of displaying closed frequent itemsets and their interrelationships) into the domain of graph theory. It is necessary to map concepts from one domain to another. If we look at figures presenting closed frequent itemsets in [2]- the itemsets are realized as vertices in the graph. Furthermore, links between itemsets can be considered as edges. The linkage was achieved in a way that certain itemsets are linked only to their supersets. In those presented visualizations every subset was linked only to one of its supersets.

In conjunction with these observations, translation from one domain to another will be achieved using the following translations:

- each closed frequent itemset (CFI) becomes a vertex
- there is an edge between each CFI and its frequent superset, edges are undirected
- weight of each edge displays the measure of additional information that is contained in different frequencies of CFI and its frequent supersets; thus weight of each edge can be defined as:

$$w(e \mid e \text{ is between vertex } A \text{ and its superset } B) = \text{frequency}(A) - \text{frequency}(B)$$

4 USING THE MINIMAL SPANNING TREE TO SHOW RELATIONSHIPS BETWEEN CLOSED FREQUENT ITEMSETS

If we would use the transformation described in previous section, and visualise it presenting all edges between all nodes – we would not be able to present it plenary without intersecting edges. Here, we introduce usage of Kruskal's algorithm to find MST of starting graph and finally get a visually acceptable structure.

4.1 Consideration of Two Typical Relations in the Construction of the Minimal Spanning Tree

There are two typical cases of relations which have to be considered together with their consequences while constructing minimal spanning tree:

I. Mutually 'chained' itemsets
II. Competitive relationship between itemsets

Ad. I. These types of relationship between itemsets are presented in Fig. 3. Here the CFI (O, P, R) is a superset of the CFI (O, P) and CFI(O). The frequency of CFI appearance is always greater than the frequency of its subset (otherwise itemsets would not be closed). Therefore, the relationship between numbers denoted in figure is the following:

$$n > 0;\ m > 0;\ k > 0 \qquad (1)$$

While forming MST, edges with minimal weights will be preferred, which results with omitting the edge between itemsets (O, P, R) and (O) in the final MST. Those itemsets are finally connected through the itemset (O, P). This will result in a structure desirable by the analyst – the structure not containing the edge weighted $m+k$ in Fig. 3. Spanning the minimal tree, as we see, ensures that no shortcuts between CFIs are presented in the final structure.

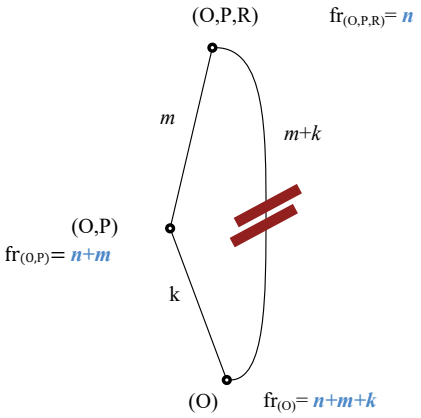

Figure 3 Mutually 'chained' itemsets

Ad. II. This situation appears when there are two different itemsets containing mutual subset that is CFI, while they also have mutual superset. These relationships would result with the cycle appearing in final graph. General example of such relationship is presented in Fig. 3. As a result of MST formation, among all edges, the one with the highest weight will be omitted in the final

structure. In Fig. 4. edges with weights *m*, *f*, *k* and *g* are presented. The length of the edges represents their weights. It should be noted that the following equality is always valid: $m+k=f+g$. The edges that exist in a graph between chain related itemsets are not documented as it is elaborated in the previous paragraph that they will safely be excluded from the final MST (i.e. the edge between (A, B) and (A, B, C, D) is not shown).

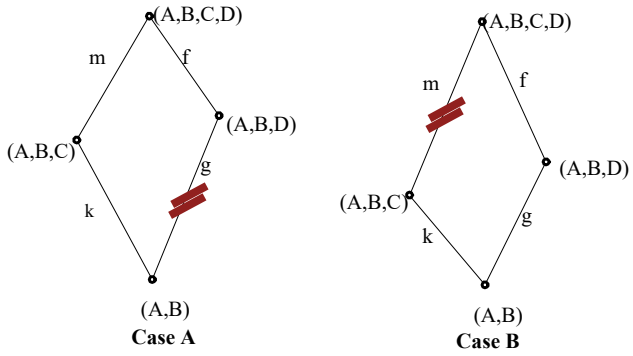

Figure 4 Competitive relationship between itemsets

With these cycle relationships, we can identify two cases that should be considered. In the case that the longest edge (with the biggest weight) is in the 'lower' part of the graph, that is, closer to the itemset with smaller number of items (case A in the figure), then the final MST is formed with edge connecting this itemset with its superset which has greater support (CFI (A,B,C) in the figure, because according to the figure $g>k$). Edge between more similar itemsets (regarding support) is kept in the final structure. Here the CFI with higher support is chosen to finally form the MST.

If the edge with the highest weight is in the 'upper' part of the graph, that is closer to the itemset with the largest number of items (case B in the figure), then the final MST is formed with edge connecting this CFI with its subset which has the lower support (CFI (A,B,D) in the figure). We can also conclude that the CFIs which are more similar according to the support parameters are connected in the final MST. That is a reasonable solution from the analyst's point of view.

4.2 Final Formation – Expanded MST

To be even more informative to the analyst, all frequent itemsets including one element, whether closed or not but a part of at least one frequent itemset, are included in the final structure. Here we somewhat deviate from the initial definition of nodes, which we find acceptable since this contributes to the overall informativeness of the final structure to the analyst. The closed one-element itemsets are added to the structure as depicted before. The other ones are included in the structure in a similar manner as the MST is built. First we take into account edges between them and all CFIs that are their supersets. The edge with the minimal weight is then chosen to form the final expanded MST structure. The example of a simple toy dataset is given in Tab. 1., and the result of our algorithm applied on it is presented in Fig. 5. This structure is much more informative than structures obtained by bottom-up and top-down strategies presented in our previous paper (Fig. 2 and Fig. 3 in the paper [2]). Finally presented information is even richer in the real world cases presented in Section 6.

Table 1 Example "toy" dataset

TID	A	B	C	D	E	F
1	1	1	0	0	0	1
2	1	1	0	0	0	0
3	1	1	0	0	0	0
4	0	0	1	1	0	0
5	0	0	1	1	0	0
6	0	0	1	1	0	0
7	1	0	0	0	1	0
8	1	1	1	0	1	0
9	1	1	0	0	1	0
10	1	1	1	1	1	0
11	0	0	0	0	0	1
12	0	0	0	0	0	1

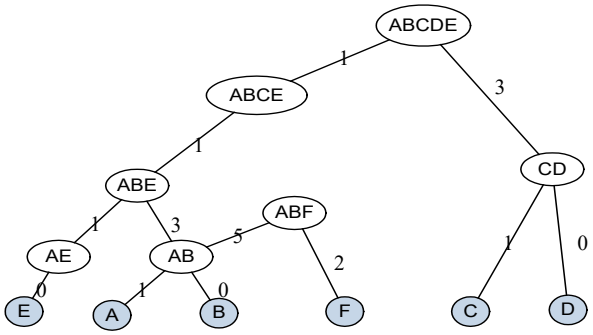

Figure 5 Kruskal's algorithm for E-MST formation

5 IMPLEMENTATION

In [2] we have described our reference tool implemented in Python programming language with a PyQt based GUI which allowed for a high-level user-friendly experience in transforming transactional datasets into dendrogram visualizations. While the tool was surprisingly effective and well-received by analysts who tested it in production environments, the choice to create a custom tool ultimately showed up to be rather restrictive and time-consuming concerning long-term management and future upgrades. To allow for easier sharing and reproducibility, as well as facilitate focusing on strategies and analytical process instead of low-level programming issues, we have decided to re-implement our solutions for analysing transactional data in R programming language, using various R packages and the R Markdown technology. This enables us to readily share our solution which is currently available at [3]. This GitHUB repository contains all the code for building graph structures described both in [2] and in this paper, together with a report explaining how to reproduce most of the visualizations shown in this paper as well as datasets used to produce them.

One drawback of our current implementation is that it expects at least an intermediate level of expertise with programming language R, especially if the analyst wants to fully leverage the capabilities of visualization engine and produce highly interpretable, report-ready visualizations. One of our immediate future goals is creation of a dashboard interface with interactive capabilities, possibly using Shiny and Plotly packages. This will facilitate the

adoption of our methods by the analysts who prefer a more hands-on approach, without the pre-requisite knowledge of R language.

6 RESULT EVALUATION

Developed Expanded MST (E-MST) method is evaluated in detail on many datasets. Here we present its performance on representative datasets and give a further insight into how it can be used in educational domain.

One advance note: some of the figures presented in this section are not necessarily representative of the experience of using the final application, as they should be seen in interactive surroundings and in larger resolution, presumably seen on a large desktop screen.

Toy dataset enables a glance at developed method performance. As Fig. 6 showcases, all frequent itemsets are present in the final structure. Additionally, the sizes of the shapes that represent the vertices depend on total support of itemset presented by the vertex. For the domain expert, this feature is very useful to gain overall impression of the role of all closed frequent itemsets.

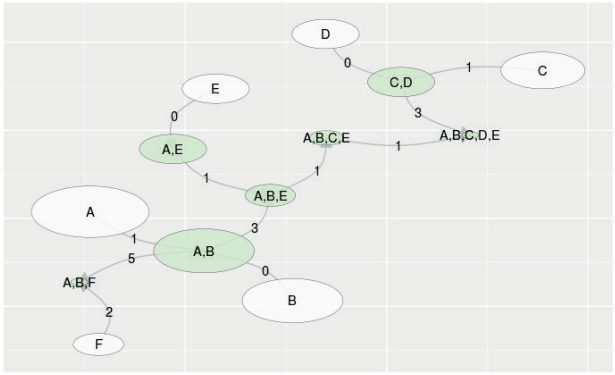

Figure 6 Toy dataset, E-MST, min. supp. 8%

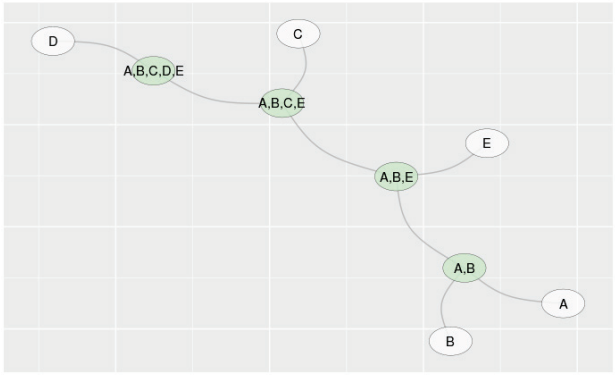

Figure 7 Toy dataset, Top-down strategy, min. supp.8%

We can see that all closed frequent itemsets are present in the final structure – which was not the case with previous strategies (top-down strategy was missing itemsets AE, CD and ABF; and the bottom-up strategy is missing itemsets AE, ABF and ABCE). For comparison, figures representing outcomes of top-down and bottom-up strategies are also given (Fig. 7 and Fig. 8).

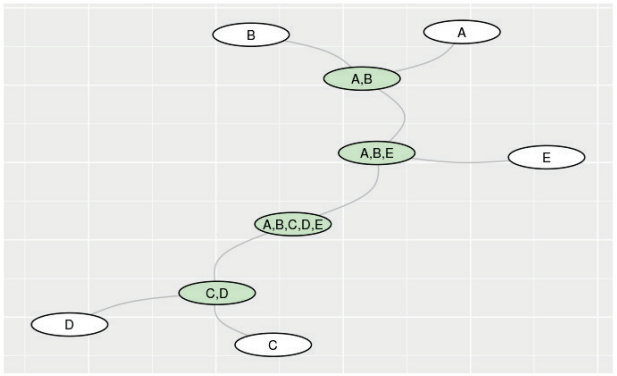

Figure 8 Toy dataset, Bottom-up strategy, min. supp.8%

ComputerShop dataset is one of the test datasets provided with the Oracle 11g database and contains transactional data dealing with sales of computer equipment. It can be considered "sparse" since it contains 940 transactions with 14 items and less than 3 items per transaction. The most frequent item occurs in 32% of all transactions.

Fig. 9 presents the result of the E-MST method, while Fig. 10 presents structures gained both by top-down and bottom-up strategies. The E-MST method reveals new relationships not seen in rather simple yet illustrative tree structures gained by bottom-up and top-down strategies. Those strategies managed to extract most important relationships (in terms of itemsets' support), but the new method adds a handful of new information while retaining the groupings shown in earlier strategies. For example, now we can see that the peripherals including "mouse pad" are also connected ("are nearer") to the group including "CD ROM" and "Pack of 5 CD-RWs". Furthermore, this group is also connected to the graphical devices: "monitor" and "graphics card".

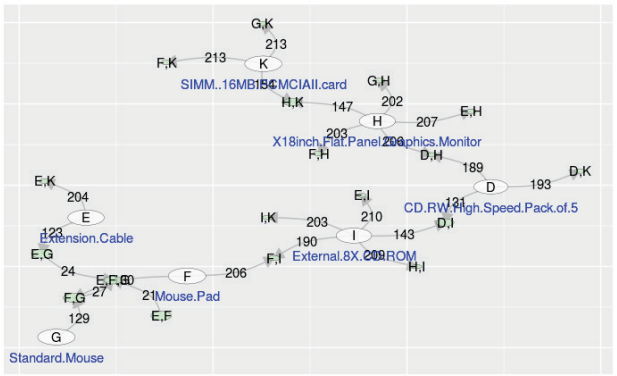

Figure 9 ComputerShop dataset, E-MST,min. supp.8%

Also, we can notice that the itemsets containing two elements appear in a star-like manner around frequent one-element items, while the numbers on edges reveal the 'distance' between frequent itemsets. We can conclude, for example, that "standard mouse" appears frequently with graphical devices (appearance of frequent itemsets GH, GK) but more rarely than with "mouse pad" (distances of 202 and 213 vs. distance 129). Similar conclusions can also be driven for certain other items.

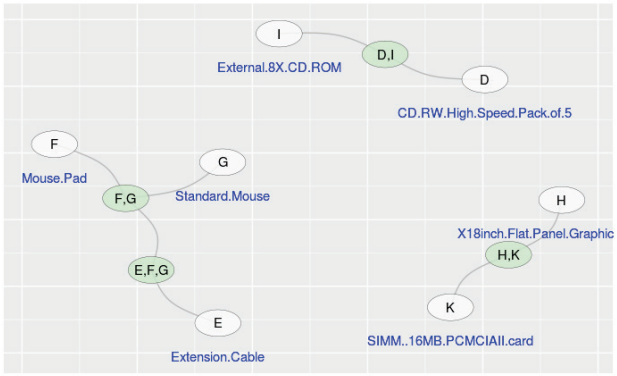

Figure 10 ComputerShop, Bottom-up and top-down str., min. supp. 8%

Educational domain is an example of a special domain where usage of E-MST can bring useful insights. [19] presents a case study where exam results are transformed into a transactional form and, in conjunction with Q-matrices (question/learning concept grid matrices) created by domain experts, can produce dendrogram structures providing insight into relationships between learning concepts. The E-MST method we propose can give further information by enriching the analysts' view on relationships between concepts. However, to get the most useful results, exam data must be transformed into transactional form in a slightly different manner.

The transformation scripts use exam results and Q-matrices to construct tables which conform to a previously chosen transactional data format. The transformation process is relatively simple: each student's exam data is observed as a transaction, containing items representing only concepts related to correctly answered questions. We also experimented with transactions where wrong answers and unanswered questions were also regarded as items of interest. This approach did not prove to be useful in this domain since these wrong answers and unanswered questions were producing many frequent itemsets which overcluttered the newly developed structures. Therefore we decided to concentrate only on concepts thus using the transformation described ahead.

To evaluate E-MST results, we performed analysis on mid-exam data gathered on the course 'Fundamentals of Electrical Engineering' which is a first year obligatory course at the University of Zagreb, Faculty of electrical engineering and computing. The course is attended by over 700 students which provided us with respectfully sized and interesting datasets. Newly developed E-MST method provided us with enriched insight into relationships between learning concepts, which can be seen in Fig. 11. Previously developed top-down strategy gave somewhat scarce information as presented by Fig. 3 in [19]. With the chosen minimal support of 20%, very interesting relationships between items presented in [19] emerge. Those relationships are presented in Fig. 12. To be even more informative, in the last two figures, we displayed the weight of edges explicitly to demonstrate the power of relationships between items i.e. learning concepts. If the difference of two connected frequent itemsets frequency is smaller the edge is drawn thicker (the actual thickness is calculated as *1/log(weight)*).

Analyst must be careful not to set the support too low, as many two-item, three-item etc. itemsets are becoming a part of the E-MST structure, masking most important relationships. Our recommendation is starting with inflated support and gradually lowering it according to analyst's wishes. Generic recommendation that would work in all cases is not possible to determine, since frequency of concepts appearing together depends on many parameters such as complexity of lectures, exam difficulty, exam structure etc.

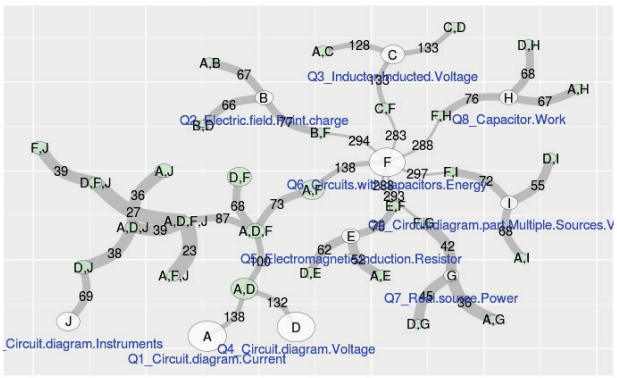

Figure11 Midexam dataset, E-MST, min. supp. 16%

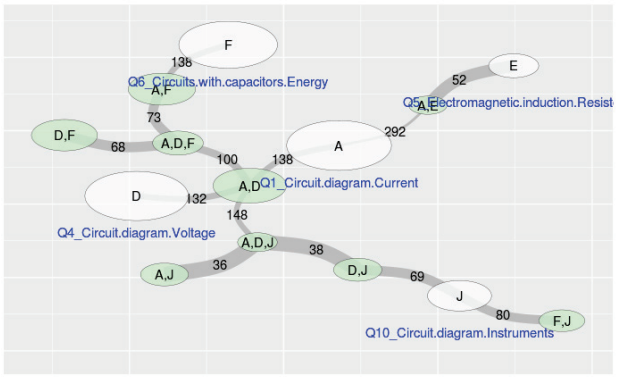

Figure 12 Midexam dataset, E-MST, min. supp. 20%

7 CONCLUSION AND FUTURE WORK

This paper presents newly developed E-MST algorithm for forming plenary graphical structures illustrating relationships appearing in transactional data. It also presents a reference application developed in R programming language which is available for usage as open source for interested parties.

The performance of described E-MST algorithm was thoroughly examined on various representative datasets where it was proved to be useful for gaining quick, overall view of relationships between items regarding their mutual appearance in transactions. Its functionality was examined in the originally intended transactional domain, on market basket data, but was also tested on educational data where we used transformed exam results to gain insight into relationships between learning concepts.

E-MST method proved to be very useful, with the examples showcasing a much richer presentation of relationships between items than with previously available strategies. However, our opinion is that the analyst should experiment with differently tailored datasets and various methods available in the application, with varying support levels and visualization options.

Future work will include integration of measures regarding gained structures, and development of application interface that would enable even easier usage of available methods. Further integration in educational domain with different datasets including data from more than one exam will be investigated together with development of larger system intended for analysis and reporting in educational domain. We also plan to make some improvements of gained structure visualizations, adding interactivity and friendlier user-interface.

Acknowledgements

The research team would like to thank Croatian Science Foundation (Hrvatska zaklada za znanost – www.hrzz.hr). The work has been fully supported by Croatian Science Foundation under the project UIP-2014-09-2051 eduMINE – Leveraging data mining methods and open technologies for enhancement of the e-learning infrastructure.

8 REFERENCES

[1] Agrawal, R., Imieliński, T., & Swami, A. (1993). Mining association rules between sets of items in large databases. *SIGMOD '93, Proceedings of the 1993 ACM SIGMOD international conference on Management of data*, Washington D.C., 207-216. https://doi.org/10.1145/170035.170072
Vranić, M., Pintar, D., & Banek, M. (2015). Towards better understanding of frequent itemset relationships through tree-like data structures. *Expert systems with applications, 42*(3), 1717-1729. https://doi.org/10.1016/j.eswa.2014.09.040

[2] Pintar, D. (2017). Implementation of transaction visualization strategies in R. *GitHUB repository*, https://github.com/ratnip/Kruskal (3.11.2017)

[3] Vo, B., Coenen, F., & Le, B. (2013). A new method for mining frequent weighted itemsets based on wit-trees. *Expert Systems with Applications, 40*(4), 1256-1264. https://doi.org/10.1016/j.eswa.2012.08.065
Deng, Z. & Lv, S. (2014). Fast mining frequent itemsets using nodesets. *Expert Systems with Applications, 41*(10), 4505-4512. https://doi.org/10.1016/j.eswa.2014.01.025

[4] Engle, K. M. & Rada, R. (2011). A top-k analysis using multi-level association rule mining for autism treatments. *UAHCI 2011, International Conference on Universal Access in Human-Computer Interaction*, Springer, Berlin, Heidelberg, 328-334. https://doi.org/10.1007/978-3-642-21657-2_35

[5] Hashem, T., Ahmed, C. F., Samiullah, M., Akther, S., Jeong, B., & Jeon, S. (2014). An efficient approach for mining cross-level closed itemsets and minimal association rules using closed itemset lattices. *Expert Systems with Applications, 41*(6), 2914-2938. https://doi.org/10.1016/j.eswa.2013.09.052

[6] Geng, L. & Hamilton, H. J. (2006). Interestingness measures for data mining: A survey. *ACM Computing Surveys, 38*(3). https://doi.org/ 10.1145/1132960.1132963

[7] Han, J. & Kamber, M. (2006). *Data mining: Concepts and techniques*. The Morgan Kaufmann series in data management systems. 2nd ed. Morgan Kaufmann, Elsevier, San Francisco.

[8] Kim, J. W. (2017). Construction and evaluation of structured association map for visual exploration of association rules. *Expert Systems with Applications*, 74, 70-81. https://doi.org/10.1016/j.eswa.2017.01.007

[9] Vranić, M., Pintar, D., & Skočir, Z. (2010). Generation and Analysis of Tree Structures Based on Association Rules and Hierarchical Clustering. *2010 Fifth International Multi-conference on Computing in the Global Information Technology*, Valencia, 48-53. https://doi.org/10.1109/ICCGI.2010.28
Pinjušić Ćuric, S., Vranić, M., & Pintar, D. (2011). Improvement of hierarchical clustering results by refinement of variable types and distance measures. *Automatika, 52*(4), 353-364. https://doi.org/10.1080/00051144.2011.11828434

[10] Pavčević, M. O. (2006). *Uvod u teoriju grafova* (Introduction to Graph Theory). Element, Zagreb (in Croatian).

[11] Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2001). *Introduction to Algorithms*. 2nd ed. MIT Press and McGraw-Hill.

[12] Nakić, I. (2011). Lecture Notes on Discrete Mathematics from 2011/12, PMF Zagreb. https://web.math.pmf.unizg.hr/nastava/komb/predavanja/predavanja.pdf (accessed 7-Nov-2017). (in Croatian)

[13] Kos, M. (2010). Lecture Notes on Information Networks from 2010/11, FER Zagreb. (in Croatian)

[14] Prim, R. C. (1957). Shortest connection networks and some generalizations. *Bell System Technical Journal, 36*(6), 1389-1401. https://doi.org/10.1002/j.1538-7305.1957.tb01515.x

[15] Kruskal, J. B. (1956). On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical Society, 7*(1), 48-50. https://doi.org/10.1090/S0002-9939-1956-0078686-7

[16] Vranić, M., Pintar, D., & Humski, L. (2016). Automated extraction and visualization of learning concept dependencies using Q-matrices and exam results. *24th International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*, Split, 1-5. https://doi.org/10.1109/SOFTCOM.2016.7772122

Contact information:

Mihaela VRANIĆ, assist. prof., first author
FER, University of Zagreb,
Unska 3, 10000 Zagreb, Croatia
Mihaela.Vranic@fer.hr

Damir PINTAR, assist. prof., corresponding author
FER, University of Zagreb,
Unska 3, 10000 Zagreb, Croatia
Damir.Pintar@fer.hr

Frano ŠKOPLJANAC-MAČINA, PhD student
FER, University of Zagreb,
Unska 3, 10000 Zagreb, Croatia
Frano.Skopljanac-Macina@fer.hr