Approach to Visualisation of Evolving Association Rule Models

Martin Hlosta, Michal Šebek, Jaroslav Zendulka, and Tomáš Hruška

Faculty of Information Technology, IT4Innovations Centre of Excellence, Brno University of Technology, Božetěchova 2, Brno, Czech Republic,
{ihlosta, isebek, zendulka, hruska}@fit.vutbr.cz

Abstract Visualization of evolving data mining models can allow good insight for a data analyst about these models and their changes in time. This paper presents our approach to visualization of evolving association rule models. This approach is based on graph visualization where nodes of the graph represent itemsets, and edges represent association rules. We show how evolving models, produced by data mining algorithms, are stored in the knowledge base, and then how they can be filtered and visualized. Two ways of graph based visualization of evolving models are shown using force based layout algorithms - local and global layout. Experiments with both are illustrated with emphasis on the latter.

Keywords: evolving models, data mining visualization, graph based visualization, force-based layout.

1 Introduction

Analysis of huge amount of data, especially finding any new knowledge about them, is a current important problem. The data mining research area is focused on solving this problem by algorithms based on principles of the artificial intelligence. The data mining process generally consists of four steps - *data preprocessing*, *data mining*, *pattern evaluation* and *knowledge presentation*. Although the main interest in data mining is focused on the data mining step, the others are also very important for successful data analysis. This paper deals with the last step of the process – knowledge presentation, specifically with the problem of association rules model visualization.

The association rule model [1] is a result of association rules data mining algorithms. Let DB be an input *transaction database* and $I = \{i_1, i_2, \ldots, i_n\}$ be a set of all items in the input database. Any subset $X \subseteq I$ is called an *itemset*. The *support of the itemse*t X is defined as a number of transactions in the database DB that contain X. Itemsets with a support greater or equal to some defined *minimal support* threshold are called *frequent itemsets*. *Association rules* are rules of the form $A \rightarrow B$, where $A, B \subset I$ and $A \cap B = \{\}$, A is called *antecedent* and B *consequent*. The *support of the association rule* $A \rightarrow B$ is defined as the support of the itemset $X = A \cup B$, the *confidence* is a fraction of transactions containing A in the database DB that contain also B. The support

is sometimes expressed in a relative form to the size of database $|DB|$. Rules that satisfy both the minimum support and minimum confidence threshold are called *strong*. A typical example of frequent itemsets and association rules mining is market basket analysis yielding the rules such as: *product A occur together B in s transactions and when customers buy product A, with probability c they buy product B.*

As new data continuously arrive, the topic of mining in data streams becomes more attractive. For mining in such datasets, it is much better to use algorithms or methods for *mining data streams*. When mining in data streams, the problem of concept drift becomes important. The concept drift was defined by Wang et. al. [2] as a problem of identifying training data that are not longer consistent with the current concepts in the actual model. We can illustrate this on the example of a shopping basket. Firstly, customers buy products A and B, then they are affected by advertising campaign, so they buy a product C instead of B. Naturally, we expect that the mining algorithm will adapt to a new behaviour of customers (new concepts) soon. However, the information that the product A was popular in a past could also be important, and we will discuss this problem in the Section 3.

Moreover, sometimes it is not desirable to analyse data set collected in several months or years as a whole. As stated above, the concept changes in time, and the objective of an analyst could be to find changes in concepts in time. In this paper, we describe the approach, which could help an analyst to reveal changes in association rules models in large data sets or data streams using *graph-based representation* of association rules and *user-based filtering of association rules* in these models.

The paper is structured as follows. Section 2 refers related work, Section 3 describes our approach to visualization of evolving association rule models. The storage of association rule models and a method of visualization are described. Section 4 discusses practical contribution to analyst using experiments on a testing dataset. Finally, Section 5 makes conclusions about using our approach to the visualization.

2 Related Work

The visualization of association rules is important for a better understanding and analysis of mined results [3]. Sometimes, the problem of the visualization in general is called a Visual Data Mining. Keim in [4] describes three main steps of Visual Data Exploration: *overview first, zoom and filter, and details-on-demand*. In [5], Keim classifies the Visual Data Mining techniques from three points of view: *data type to be visualized*, the *visualization technique*, and the *interaction and distortion technique*.

We focus on the visualization of association rules. Basic principles of association rules visualization are summarized in [6] and [7]. They divide the problem of the association visualization into four cases: *one-to-one* (both antecedent and consequent contain only one item), *many-to-one* (antecedent contains more

items, consequent only one item), *one-to-many* (one item in antecedent, more items in consequent) and *many-to-many* (both antecedent and consequent contain more that one item). Here we mention three most common methods.

Rule Table – is the basic method to a visual representation of association rules. The method displays association rule attributes in particular columns: antecedent, consequent, support and confidence. An advantage is a possibility to sort rules by each column. It enables to visualize many-to-many associations. A disadvantage is that it is not possible to display that some item is shared by several rules. The Rule Table was implemented for example in the tool of Chakravarthy et. al. described in [8].

Two-Dimensional Matrix – displays antecedent and consequent on X- and Y-axes. Values on intersections represent the strength of rules (support or confidence). Hofmann in his modification [9] used to express the support and confidence of a rule colours and tones. The two-dimensional makes it possible to visualize only matrix only of one-to-one associations.

Directed-Graph – was introduced in [10]. The principle is to display association rules as a hyper-graph where nodes are items, and edges consist of all items in the rule. Many current data mining systems use modifications of this visualization method. For example, DBMiner [11] uses a directed hyper-graph (many-to-many associations) or MS Business Intelligence Development Studio [1], which uses a classical directed graph (one-to-one associations only). The big advantage of this method is that relationships between items and itemsets are clear and understandable. On the other hand, the approach is not usable for many association rules. Then it becomes confusing, and the number of association rules should be reduced by filters.

To place nodes of the graph in an intelligent way, there exist several methods of computing a layout of a graph. A specific type of method should be chosen based on the application area and characteristics of the graph. For association rule models, force-based algorithms are the most suitable choice. They model the graph as a physical system with attraction and repulsion forces among its nodes. Repulsive forces act among each pair of the nodes, attraction forces act only among those nodes that are connected by edges. When computing a layout, force-based algorithms try to find the global minimum of energy of this model, typically using the gradient descent method.

There exist various force-based models that differ in a way, how forces among nodes are defined, which results in various graph layouts. One of the first models, *Kamada-Kawai (KK)* [12] model, considered only attraction forces among nodes analogous to spring forces. This is suitable only for one-component graphs. To support graph with more than one component, repulsive forces should be included. *Fruchterman-Reingold (FR)* [13] employs repulsive force analogous to electromagnetic repulsion. In the *LinLog* model by Noack [14], a repulsion force was modified in such way that a logarithm of a distance between nodes is computed instead of a square. This change causes that separate clusters occur in

[1] MS Business Intelligence Development Studio: `http://msdn.microsoft.com/en-us/library/ms173767(v=sql.105).aspx`

the graph layout representing areas with densely connected nodes. In addition, this model utilizes weights of edges where nodes connected by edges with higher weights are attracted more than those with lower weights. Besides force-driven layouts, it's worth mentioning spectral, orthogonal, tree, layered (*Sugyiama*) and circular layout methods.

There are some other ways how the association rules can be visualized: methods based on Mosaic plots [15], the visualization in parallel coordinates [16,17] or methods based on combinations of described approaches [18].

There are some extending research works that discuss the visualization of evolving (active) data mining models. This means that models are not static during the time, and they focus on the visualization of changes in models. In [6] Ong et. al. proposed the colour-based method for visualization of *changes of models* in the Grid View visualization models.

3 Analysing evolving association rule models

Our approach to analysing evolving association rule models is shown on Fig. 1. The input is a *data stream* or a *large dataset*. In our approach, the input is split into mining windows, typically based on non-overlapping time intervals. The whole approach is based on three basic components. The *Mining Algorithm* for mining association rules is run for each *Mining Window* for the data mining task. The results of runs of the Mining Algorithm are saved into the storage component called the *Knowledge Base*. Because models are results of different data mining task, Knowledge Base supports categorization of models into the *Knowledge Groups*. Finally, association rule models are restored from the Knowledge Base, and they are presented to an analyst in a component called the Presenter in a form of *Evolving Models*. The Presenter provides loading of the specified models using the user-specified restrictions containing an identification of the task, time interval settings and filtering based on items in association rules. The main contribution of this approach is that it simplifies the analysis the concept drift in the input data in the visual form of association rule models.

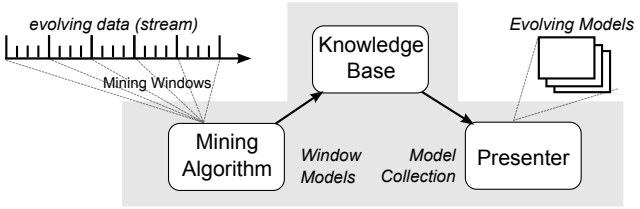


Figure 1: Approach to visualization of evolving association rule models.

3.1 Knowledge Base

The storage component in our approach of analysing evolving models is the Knowledge Base. Although the paper deals with association rule models, the Knowledge Base is designed generally for storing of any kind of evolving models.

Models in Knowledge Base are classified into containers called *Knowledge Group*s. Groups are identified by a human-readable string label. Typically, a group contains models mined by one task in different mining windows. The core of a saved structure is formed by a series of time-ordered models. Models are filterable by an identification of the group and by the time restrictions. Time representation of models in a form of a *valid time*, which is a time interval when the model is valid.

For physical representation of *model data* the XML[2] based document is used. The main benefit is that the XML is well-structured and supported by current database systems. Our XML for representation of association rule models enables also an extended filtering over saved models using XPath. The XML document format is inspired by the standard PMML[3] representation. However, PMML is not suitable to be used as internal representation in the Knowledge Base because the filtering inside association rule models would be very complicated and time consuming. Our Knowledge Base format uses redundant, but the XPath more efficient filtering format, where nodes directly contain inner nodes instead of using references (e.g. node *Itemset* contain one ore more nodes *Item*).

3.2 Presenter

The Evolving AR Presenter is a tool for visualization of evolving association rule models. For better demonstration of our approach, we describe the work with the presenter tool as a process, which is depicted in the Fig. 2.

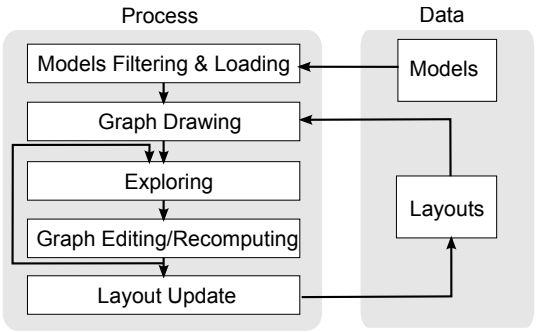


Figure 2: The process of visualization evolving association rule models using the presenter tool.

[2] Extensible Markup Language (XML): http://www.w3.org/XML/
[3] PMML Standard: http://www.dmg.org/v4-1/GeneralStructure.html

As mentioned above, the source evolving association rule models are stored in the Knowledge Base. Firstly, the user selects Knowledge Group to be analysed. For selected group, the user sets filtering parameters by means of *filtering configurations panel*. This can simplify visual analysis when the number of rules is very huge.

The *filtering configurations panel* allows the user to select a previously stored named filtering configuration or to create a new one by providing some filtering constraints. These can be applied to association rule models of the selected Knowledge Group as a whole or to the association rules it contains. The former enables to specify time constraints on valid time windows. With the latter, it is possible to set constraints for the rules (e.g. the minimum support or confidence threshold) or constraints over antecedent or consequent (e.g. antecedent contains "TV").

After the specification of filtering constraints, association rule models are asynchronously loaded from the Knowledge Base, and they are ready for an exploration. The exploration view is divided into three areas that are depicted on the screenshot of the application on the Fig. 3: the selected model visualization, the model moving control, and the rules filtering sliders.

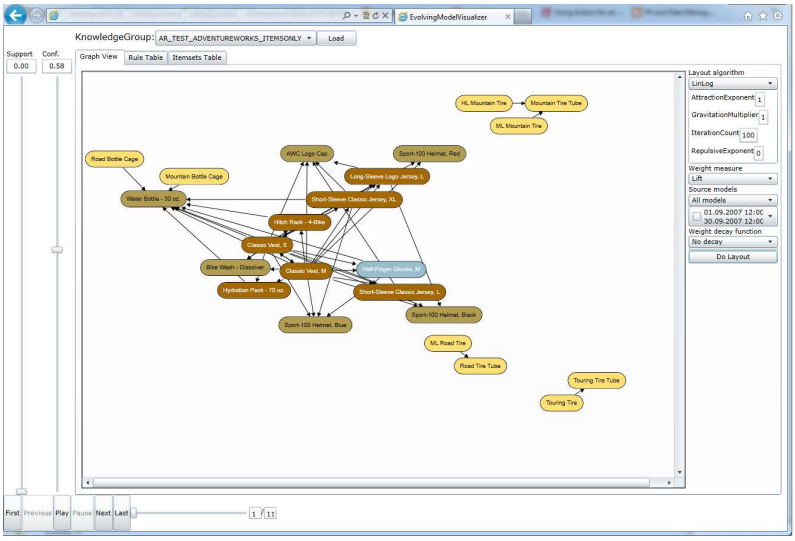


Figure 3: Presenter window - selected association rule model view and controls for changing the actual model and filtering.

The greatest area covers the view of the currently selected model from the Knowledge Group. The tool supports three types of views, graph based view, rules table and itemsets table view. The graph based view represents the key part of the visualization process, and it will be described in more detail in the

following paragraph. The remaining two views contain tables of itemsets and rules with couple of interesting measures – support for itemsets; support, confidence, lift, leverage, and affinity for association rules. The measures comply with ones contained in the association rule model defined in the most recent version of PMML 4.1[4].

The bottom part serves for changing the selected model by either moving forward or backward in all the available models of a knowledge group. Using filtering sliders on the right, the user can filter rules and frequent patterns in the selected model by increasing/decreasing support and confidence threshold.

3.3 Graph view and graph based visualization

The key part of our approach is based on a directed-graph representation. Nodes of the graphs represent single items or whole itemsets connected by edges representing rules. This decision of using items or itemsets depends on an user needs.

For a filtering configuration created in the past, stored positions of nodes are loaded. Otherwise, the layout of the graph is computed by one of the layout algorithms with specified parameters. After drawing the graph, the user can manually adjust these positions, and they are automatically stored for any analysis in the future. For better clarity, the change of the current model with respect to the model in the previous window is treated as follows. Nodes that were added in the currently viewed model are emphasized by green colour. Nodes that were removed in comparison to previous model, and thus they are not in the current model, are also drawn but with increased transparency level.

The goal we wanted to achieve was to automatically draw a layout, where related nodes occur together and closer than those that are less related. These closely related nodes should form well separated clusters from the other nodes or clusters of nodes. To achieve this goal, the graph layout algorithms KK, FR, and LinLog were chosen for visualization. The limitation of the KK algorithm is the recommendation to use them only for well connected graphs. Association rule models typically result in more graph components. Next, thanks to LinLog primary purpose to reveal clusters in the graphs and its support for weighted edges, we assumed that this algorithm will produce the best layout.

LinLog layout model utilizes weights of the edges. As mentioned above, the presenter provides for that purpose usage of interesting measures from the association rule model in PMML 4.1 specification, which are

- *support*, *confidence* – defined in Section 1,
- *lift* – $lift(A \to C) = \frac{confidence(A \to C)}{support(C)}$,
- *affinity* – $affinity(A \to C) = \frac{support(A,C)}{support(A)+support(C)-support(A,C)}$ and
- *leverage* – $leverage(A \to C) = support(A \to C) - support(A) \cdot support(C)$, which in contrast to previous two measures emphasizes the *support* of common occurrence of A and C.

[4] PMML 4.1 Association Rule model: http://dmg.org/v4-1/AssociationRules.html

The choice of used measure is up to an user, how he wants to emphasize the bounds between nodes.

When dealing with evolving association rule model, the important task is not only to provide good layout for one of its partial models but for all of them and also to deal with changes between the partial models. There exist two possibilities of how to draw evolving association rule models.

1. *Separate layout for each model* - this means, that the layout algorithm is recomputed for each of the models. This leads to different nodes positions for each of them. The advantage is that each layout is well adjusted for the current model, and its size is smaller because there isn't reserved empty space for nodes in different models. However, the user loses some information about the association rules in the global perspective. Moreover, the change of positions of some nodes can make it more difficult for the user to understand how the clusters are changing, and how nodes were added or removed from the current model.
2. *One global layout over all models* - instead of computing one layout for each model, only one layout for all models is created. Opposed to first approach, it has one global view in all models with the cost of increased size because of potential blank spaces, where these nodes doesn't occur in some models.

The presenter tool supports both the possibilities. Here, we primarily focus on the second one. The question is, how to deal with more association rule models when drawing the layout. For layout algorithm, which doesn't support weights for edges, the solution is straightforward. If two nodes are connected in any of the partial models, they will be connected in the graph, from which the global layout is computed. We refer to this graph as a *global graph.* For LinLog, which supports weights, the question is, how to set the edge in a global graph. We propose two possibilities.

1. Every partial model is considered *equal to another*. Then, an edge weight in a global layout, $weight_{glob}(e)$, is computed as an average of weight over all its models:
$$weight_{glob}(e) = \frac{\sum_{1 \leq i \leq n} weight_i(e)}{n}, \tag{1}$$
where n is the number of models, $weight_i(e)$ is the weight of the edge e in the $i-th$ model based on selected interesting measure. If the edge e is not in the $i-th$ model, then $weight_i(e) = 0$.
2. Some models are assigned higher importance than the others. Typically, older models can be less important than the recent ones. To emphasize the importance of more recent edges, we provide the possibility to apply a decay function on weights of edges. Usually, for that purpose, exponential decay is used, which is available in the presenter together with linear decay. The global weight of the edge e is then computed as a weighted average weighted by the value of the decay function for the model, $weight_{dec}(i)$ for $i-th$ model,
$$weight_{glob}(e) = \frac{\sum_{1 \leq i \leq n} weight_{dec}(i) \times weight_i(e)}{\sum_{1 \leq i \leq n} weight_{dec}(i)}. \tag{2}$$

The implementation of the layout algorithms in Graph# library [5] for WPF was used together with its modification for Silverlight [6]. In addition, the *Force-Scan algorithm* from Graph# library was used to remove the overlapping between nodes.

4 Experiments

In this section, we demonstrate the practical contribution of our approach on the sample dataset AdventureWorks[7], available for Microsoft SQL Server. The dataset contains data about market trade transactions. FP-tree algorithm was used for mining association rules. The dataset was divided into 11 mining windows of 1 month length, which were analysed (mined) separately. The parameters of the mining process were: *minimum support = 44, minimum confidence = 0.4,* the *maximal itemset* size was limited to *2.* Statistics about data set windows, and mined association rules are shown in Table 1.

Window ID	1	2	3	4	5	6	7	8	9	10	11
Transactions Count	1 783	1 779	1 889	2 272	1 946	2 032	2 109	2 128	2 386	2 374	976
Items Count	187	167	173	169	145	151	146	170	170	166	42
Assoc. Rules Count	117	10	27	22	9	11	8	10	23	20	6

Table 1: Statistics of the AdventureWorks dataset mining windows.

The first experiment is focused on demonstrating the differences of layouts generated by KK, FR, and LinLog layout algorithms. Results are shown on Fig. 4. Although the figures has a small size, the differences in layouts, which we want to show, are clear. For better clarity, we show only two of the algorithms - KK and LinLog. In the KK layout, there are visible overlapping components and thus it's weakness when drawing layout graphs with more components. The LinLog layout algorithm was chosen for following experiments because it can be seen that it creates related nodes naturally clustered, and it utilizes weights of edges. Remember that for better observing the changes in association rule models, our tool uses colours to emphasize some special states of nodes.

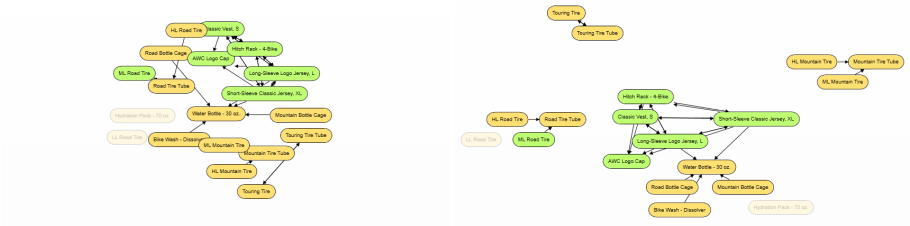
The second experiment compares two possibilities for building layout presented in Section 3.2. The first possibility is to create0 separate layouts. The case is shown on Fig. 5. You can see that if layouts are created separately, positions of nodes differ in layouts for different mining windows. Therefore, the observation of changes can be difficult.

However, second case, which uses a global layout for all the window models, keeps the positions of nodes on the same place for all the mining windows. The

[5] Graph# library: `http://graphsharp.codeplex.com/`

[6] Graph# Silverlight wrapper. `http://dl.dropbox.com/u/1355498/GraphSharp/GraphSharp.Silverlight.7z`

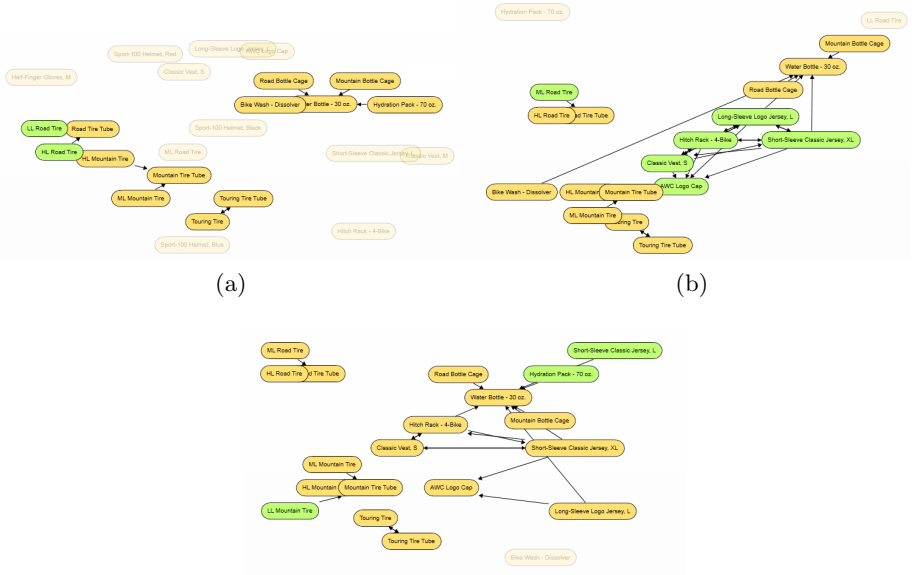
[7] Dataset AdventureWorks. `http://msftdbprodsamples.codeplex.com/`



(a) (b)

Figure 4: Comparison of layouts of association rule models produced by (a) KK and (b) LinLog algorithms.

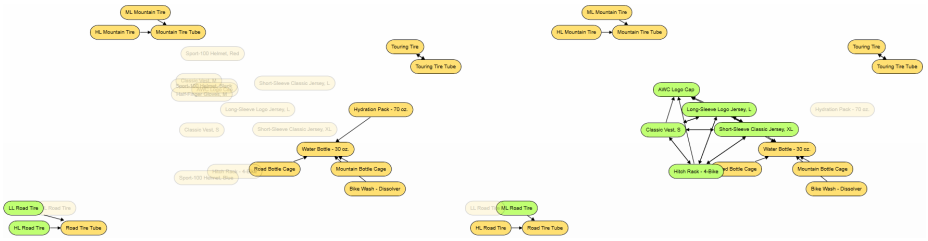
example situation for the same models as in the previous example is shown on Fig. 6. The analyst can easily observe the change between models because he can focus on tracking changes – new or removed nodes.



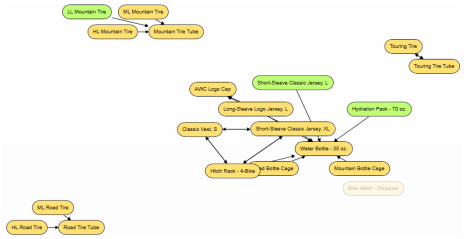
(a) (b)

(c)

Figure 5: Subfigures (a), (b), (c) show a series of association rule models where a new layout is created for each mining window.



(a) (b)



(c)

Figure 6: Subfigures (a), (b), (c) show a gradual change in association rule models where layout is global over all mining window models. This method enables to easily observe changes between models.

5 Conclusion

The visualization is very important part of the data mining that can help the analyst to understand better the mined result. We focused on visualization of evolving association rules represented by a sequence of models discovered in subsequent time windows of a large data set or a data stream. LinLog layout algorithm and a global layout seem to be the most promising techniques.

In the future, we want to incorporate semantic knowledge about the association rule models into visualization and to improve working with the clusters of nodes.

Acknowledgement

This work has been supported by the research programme TAČR TA01010858, BUT FIT grant FIT-S-11-2, by the research plan MSM 0021630528, and the European Regional Development Fund in the IT4Innovations Centre of Excellence project (CZ.1.05/1.1.00/02.0070).

References

1. Agrawal, R., Imieliński, T., Swami, A.: Mining association rules between sets of items in large databases. SIGMOD Rec. **22**(2) (June 1993) 207–216
2. Wang, H., Fan, W., Yu, P.S., Han, J.: Mining concept-drifting data streams using ensemble classifiers. In: Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining. KDD '03, New York, NY, USA, ACM (2003) 226–235
3. Jiang, N., Gruenwald, L.: Research issues in data stream association rule mining. SIGMOD Rec. **35**(1) (March 2006) 14–19
4. Keim, D.A.: Information visualization and visual data mining. Visualization and Computer Graphics, IEEE Transactions on **8**(1) (2002) 1 –8
5. Keim, D.A.: Visual exploration of large data sets. Commun. ACM **44**(8) (August 2001) 38–44
6. Ong, H.H., Ong, K.L., Ng, W.K., LIM, E.P.: Crystalclear: Active visualization of association rules. In: Conjunction with the IEEE International Conference on Data Mining (ICDM 2002). International Workshop on Active Mining (2002)
7. Bruzzese, D., Davino, C.: Visual mining of association rules. In Simoff, S., Böhlen, M., Mazeika, A., eds.: Visual Data Mining. Volume 4404 of Lecture Notes in Computer Science. Springer Berlin / Heidelberg (2008) 103–122
8. Chakravarthy, S., Zhang, H.: Visualization of association rules over relational dbmss. In: Proceedings of the 2003 ACM symposium on Applied computing. SAC '03, New York, NY, USA, ACM (2003) 922–926
9. Hofmann, H., Wilhelm, A.: Visual comparison of association rules. Computational Statistics **26**(3) (2001) 399–415
10. Klemettinen, M., Mannila, H., Ronkainen, P., Toivonen, H., Verkamo, A.I.: Finding interesting rules from large sets of discovered association rules. In: Proceedings of the third international conference on Information and knowledge management. CIKM '94, New York, NY, USA, ACM (1994) 401–407
11. Han, J., Chiang, J.Y., Chee, S., Chen, J., et. al.: Dbminer: a system for data mining in relational databases and data warehouses. In: Proceedings of the 1997 conference of the Centre for Advanced Studies on Collaborative research. CASCON '97, IBM Press (1997) 249–260
12. Kamada, T., Kawai, S.: An algorithm for drawing general undirected graphs. Inf. Process. Lett. **31**(1) (April 1989) 7–15
13. Fruchterman, T.M.J., Reingold, E.M.: Graph drawing by force-directed placement. Softw. Pract. Exper. **21**(11) (November 1991) 1129–1164
14. Noack, A.: Energy models for graph clustering. Journal of Graph Algorithms and Applications **11**(2) (2007) 453–480
15. Hofmann, H., Siebes, A.P.J.M., Wilhelm, A.F.X.: Visualizing association rules with interactive mosaic plots. In: Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining. KDD '00, New York, NY, USA, ACM (2000) 227–235
16. Bruzzese, D., Davino, C.: Visualpost-analysis of associationrules. Journal of Visual Languages & Computing **14**(6) (2003) 621–635
17. Yang, L.: Pruning and visualizing generalized association rules in parallel coordinates. Knowledge and Data Engineering, IEEE Transactions on **17**(1) (2005) 60 – 70
18. Bruzzese, D., Buono, P.: Combining visual techniques for association rules exploration. In: Proceedings of the working conference on Advanced visual interfaces. AVI '04, New York, NY, USA, ACM (2004) 381–384