Automatic Predictor Generator & Behaviour Rule Extractor – A System Proposal

Michal Puheim[1], Ján Paralič[2], Ladislav Madarász[3]
Department of Cybernetics and Artificial Intelligence
Technical University of Kosice
Letná 9, 042 00 Košice, Slovak Republic
[1]michal.puheim@tuke.sk, [2]jan.paralic@tuke.sk, [3]ladislav.madarasz@tuke.sk

***Abstract*—In this paper we present a proposal for a data-mining system deployed as a cloud service which is supposed to be used for a big data analysis. The main purpose of the system is the analysis of a vast number of event logs using means of data aggregation, clustering, classification and prediction. The system is composed of two components implemented as software services. The Automatic Predictor Generator is supposed to provide a meaningful way to aggregate large amounts of data and the Automatic Behavior Rule Extractor deals with proper analysis of these aggregations. Results of the system are the prediction rules usable for support of decision-making and in areas such as management, marketing, customer segmentation, classification, behavior prediction etc.**

Keywords—big data; data analytics; data mining; prediction; data aggregation; rule extraction; cloud services

I. INTRODUCTION

The technological progression in the last decades has led to a vast influx of data from various distinctive domains. The idiom *big data* was established to describe this emerging trend in data acquisition and processing [1]. Big data influences our everyday lives with applications including large-scale scientific experiments, health care systems, data storage, social data-mining, logistics and delivery services, etc. Main problems of big data are their vast volume, varied structure and resulting problems with its processing and real-time analysis. The amount of processed data has reached the exascale, where a single analytic system processes up to 10^{18} calculations per second [2]. It is obvious that this development requires new system architectures for data acquisition, transfer, storage and processing. Growing number of data processing applications and services begin to rely on layered system architectures (see Fig. 1) and cloud infrastructure in order to manage such enormous amounts of data [2].

The rest of this paper is organized as follows: Remaining parts of the opening section will be oriented towards the presentation of the research project [3] under which this paper is prepared. After that we will introduce the basic concepts and methodologies used in later sections. Second section deals with the actual system proposal and the third section presents the basic information about technologies used for its implementation. In the last section we present some preliminary results of the system along with our next goals and plans for the future.

A. Overview of the Research Area

With the implementation of the project [3] we focus on applied research and development with goal of providing an efficient software solution as a supporting tool for various processes in different domains of big data analysis. We aim to create a set of software services which could adapt to the specific needs and requirements of the target user [4]. Our tools and services will support business processes such as sales, marketing, personalization and recommendations, risk management, optimization of production processes, health care improvement, etc. [3][4]. One of the results of our research will be an implementation of a software platform for analysis and optimization of processes provided in form of cloud-based *software as a service* solution. Our research team is divided into several groups which deal with:

- big textual data analysis,
- aspect-based sentiment analysis in documents,
- analysis of medical data for improvement of diagnostic and other processes in health care provision,
- process and event log mining; optimization of sales and marketing processes; personalized recommendations engines.

Within the last group we deal with behavior analysis of customers and IT portal users based on event and access logs (like access to the IT portal, participation within the email campaign, display of the product in e-shop, etc.), with aim to extract various knowledge: classification rules, segmentation and clustering based on similar behavior, creation of recommendations, behavior pattern discovery. We work on tools and services that could be used for example to analyze the typical use of IT portal application with aim to optimize the user interface by simplifying the access to the mostly used controls.

Another use case of these tools is tied with the marketing analysis scenarios, such as the discovery of customers who have the potential to buy specific product after a deliberate email campaign. Various other scenarios are possible, mostly related to the customer segmentation, classification and behavior prediction. The system proposed in this paper is supposed to deal specifically with such problems.

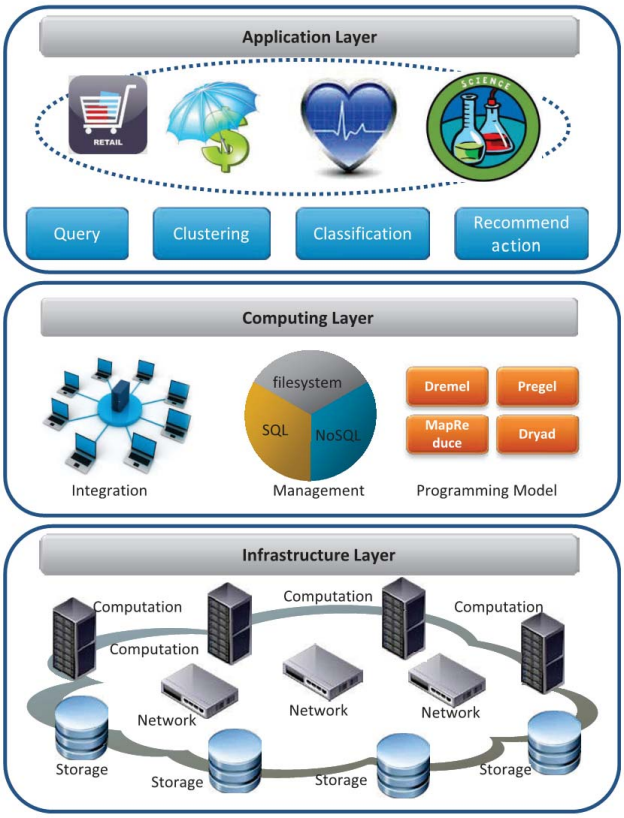

Fig. 1. Layered architecture of big data system. Source: [1]

B. *Definition of the "Predictor" Concept*

As it was outlined in the previous section, according to the philosophy of big data, we expect a database containing logs or *events* which may update and grow constantly. It is also assumed, that each *event* is tied to a corresponding person or an *entity*, which is responsible for its creation. Therefore we expect a semi-structured database which contains at least two tables – an *event table* and an *entity table* (although it is possible to use several various event tables at the same time). Each row in these tables represents a *single event* or a *single entity*; however *several events* may have a relation to only *one entity*. Naturally, each event or entity has a *set of attributes* which are supposed to be the object of further data mining analysis. It is obvious that the *rate of growth* in the size of the event table(s) is going to be *unfeasible* for the direct analysis of the event attributes. Therefore it is necessary to first *aggregate* these *numerous event attributes* into a *single entity attribute*, which can be later used for further processing.

Since the number of *available aggregation functions* as well as the number of *resulting aggregated attributes* is very high (theoretically *infinite*), our goal is to create a method which will be able to identify only the *most important* aggregated attributes (or *key attributes*) which have influence over the selected *target attribute*. For an example, we may identify the *number of purchases* as the *key attribute* and the *probability of buying another product* as the *target attribute*. We refer to the *aggregated key attribute* as *predictor* [5].

We consider that each attribute can be either of *numeric* or of *nominal* type. Therefore the influence of a single predictor to its target attribute can be evaluated using selected *correlation coefficient* for numeric attributes or *chi-square test* for nominal attributes. The calculated influence can be used in order to estimate and predict the target attribute value in the future. In general, the greater is the influence, the better is the predictor. However, for proper prediction (or classification, etc.) it is viable to generate several predictors, with as low mutual correlation as possible, because predictors with high mutual correlation can be considered redundant.

C. *Prediction and Rule Extraction*

Aggregated attributes or predictors can be further used as input attributes in order to create prediction or classification models. In order to build such models, we can employ various machine learning algorithms, such as neural networks, regression models, decision trees, support vector machines etc. As an example we can consider a decision tree which can classify existing customers into two groups: "customers who will *leave*" and "customers who will *stay*. In the tree structure, leaves represent class labels (leave/stay) and branches specify the order of coupling of features that lead to those class labels. This decision tree can be considered a predictive model which maps observations about a customer to conclusions about his target attribute. In other way, the model is predicting the future behavior of a customer. [6]

The advantage of the decision tree over other methods is that it can be used to explicitly represent and visualize decisions and hence help in decision analysis. Another benefit is relatively simple extraction of rules from the model in the form:

IF *attr_1_value* AND … AND *attr_n_value* THEN *class*

As stated by [7] there are at least three arguments in favor of rule extraction. According to the first argument, production rules are a well-known and widely spread means for representing knowledge within expert systems. Secondly, production rules are more modular and easier to understand than original decision trees. Finally, and most importantly, rules provide better generalization in resulting models and allow combination of different decision trees for the same task conforming the big data philosophy.

II. System Proposal

In this section and its subsections we provide a detailed description of the proposed system and its underlying components, which are the *Automatic Predictor Generator* and the *Automatic Behavior Rules Extractor*. Further we present the service oriented interconnection of these components and its integration to the planned cloud platform which is being implemented within the ongoing project [3].

A. *Automatic Predictor Generator*

The goal of the *Automatic Predictor Generator* (APG) is to generate specified number of meaningful *predictors* for a given *target attribute* using one or more *event tables*

and use these predictors to populate the *table of entities*. The simplification of this process is depicted on Fig. 2.

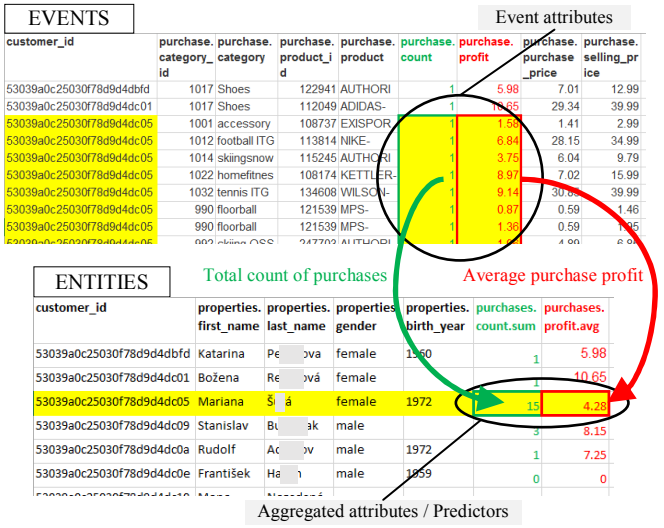

Fig. 2. Basic principle of APG operation.

Fig. 3 depicts a general algorithm of the APG and it is followed by a detailed description of the system operation:

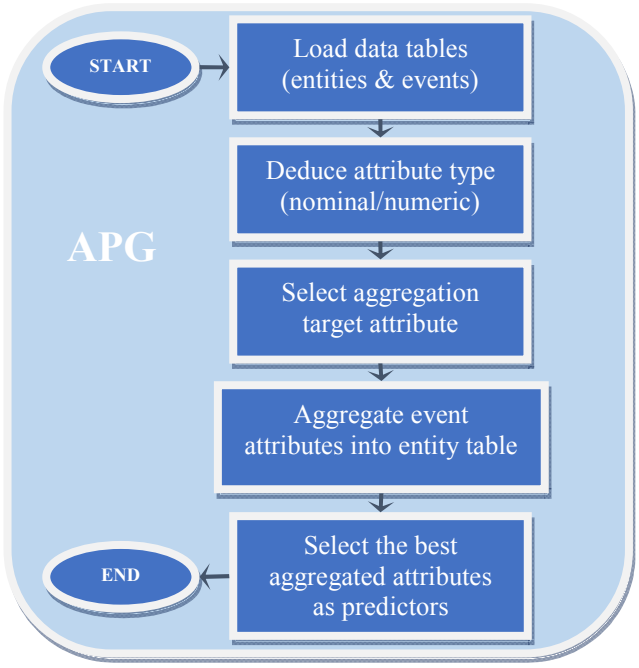

Fig. 3. General algorithm of the APG method.

Initially the *datatypes* of both *event attributes* and *entity attributes* are detected. The detection process is currently quite simple: Attributes which can be converted to numbers are considered *numeric*; all others are supposed to be *nominal*.

Next, the list of initial entity attributes is stored (i.e. the *safe list*), the prediction *target attribute* is selected and the predictor generation process is initiated. This process consists of two stages:

In the first stage, the new attributes are aggregated and added to the *entity table*. The following operators are used for the aggregation:

- *count*, *sum*, *average*, *max*, *min* and *variance* for numeric event attributes;
- *count*, *unique count*, *favorite frequency* and *favorite item* for nominal event attributes.

At present we use all of the event attributes and operators for aggregation, however this can be tweaked in the future for better performance. After the generation we get a total of *nominal attribute count* * *nominal operator count* + *numeric attribute count* * *numeric operator count* newly aggregated attributes, which are added to the entity table.

These aggregated attributes are filtered in the second stage with goal of choosing only the specified number of best predictors. This process starts with calculation of correlation coefficients between the aggregated attributes and the target attribute. The correlation coefficients are calculated according to:

- *Pearson's product-moment correlation coefficient* for numeric attributes
- *Pearson's chi-square test of independence* for nominal attributes.

In case of comparing two attributes of different type, the numeric attribute is temporarily converted to a nominal attribute (using discretization to a number of levels equal to the namespace size of the compared nominal attribute) and the chi-square test is performed. If the test fails for any reason (e.g. if too many expected frequencies are below 5), the result of the test is automatically considered zero.

Afterwards, the attributes are clustered and filtered using the principles of *Hierarchical Agglomerative Clustering* (HAC) [8][9] in the following manner:

1. All of the attributes are sorted by their correlation towards the target attribute.
2. Attributes which have their correlation coefficient lower than 1/10 of the *correlation of the best attribute* are removed (this increases overall performance of further clustering).
3. The remaining attributes are clustered using HAC where the metric used to determine the cluster distance list is the mutual correlation of attributes.
4. The resulting count of clusters equals the number of required predictors.
5. Predictors are selected as the best attributes in each cluster (i.e. those having the best correlation with the target attribute).

Finally all the *failed attributes* are removed from the *entity table* (except for those within the *safe list*) and the *updated entity table* is returned as the result of the APG method.

B. *Automatic Behavior Rule Extractor*

The resulting *entity table* modified by the APG is used within the *Automatic Behavior Rule Extractor* (ABRE) for further analysis. The result of the ABRE is a set of the most significant rules which apply to the target attribute. The algorithm of the ABRE method is shown on the Fig. 4 and the description follows the figure.

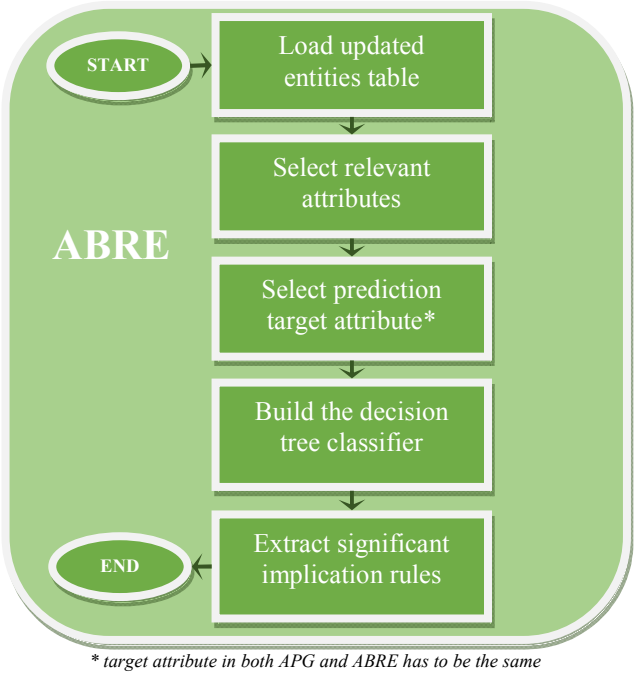

Fig. 4. General algorithm of the ABRE method

Initially, only the relevant entity attributes are selected. Irrelevant attributes with low information value, such as *entity IDs*, *names, surnames* etc. can be omitted. After that the target attribute (class label) is selected. The selected attribute needs to be the same as the target attribute used within the APG. The attribute should be nominal, however numeric attribute can be used after discretization.

Afterwards, the decision tree is created using selected tree generation technique. In our case the tree is generated using the standard Recursive Partitioning [10], which is recursively splitting the values of attributes. In every recursion the algorithm selects an attribute to split using a specified selection criterion, in our case the Gini Index [11]. The attribute is split according to the count of unique values of the attribute thus producing a tree with one branch for each unique value. Each branch has a descendant subtree (or a label value) produced by a recursive application of the same algorithm. The recursion is stopped when either of the following conditions applies:

- All (or vast majority) of the examples of an attribute have the same label value.
- There is only a small number or examples remaining for the current subtree.
- The maximum specified tree depth is reached.

After the generation, the tree may get through a pruning procedure, where insignificant leaf nodes of the tree are removed. This often helps to prevent over-fitting and to increase the predictive power on unseen datasets.

Finally a set of rules is extracted from the decision tree. This set is sorted according to the number of examples to which a single rule applies correctly. Only the specified number of most significant rules is selected.

C. *Combined System as Service*

Our plan is to integrate the proposed methods as the cloud services within the implementation of the project [3]. Therefore we consider the interfaces with the methods being message oriented, see Fig. 5.

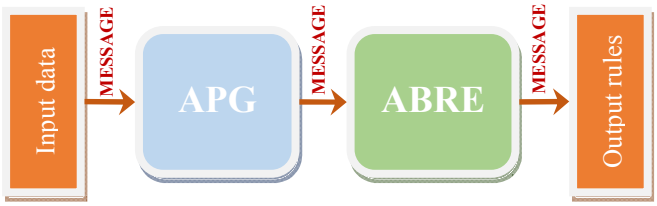

Fig. 5. Proposed service implementation of APG and ABRE.

We also aim to parallelize the methods and run them on several virtual clusters (or containers) at once. The scheme of proposed network architecture of the parallelized APG and ABRE systems is shown on Fig. 6.

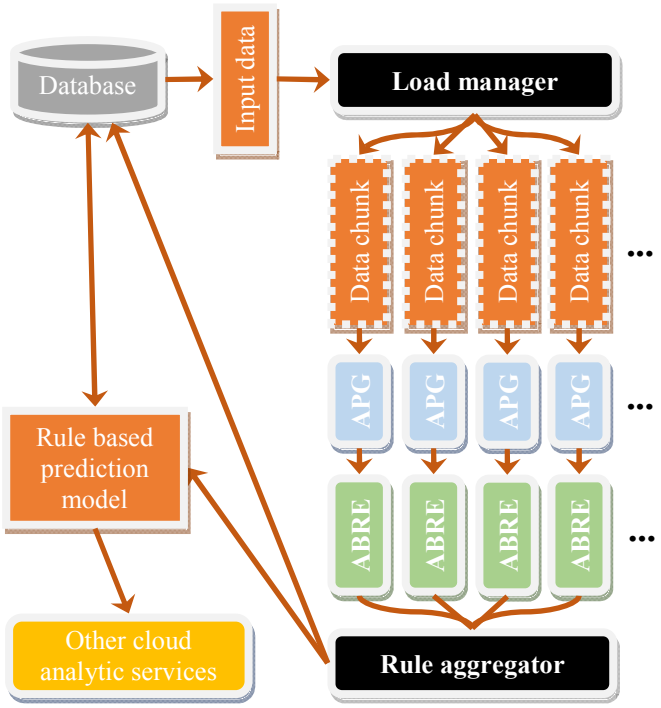

Fig. 6. APG & ABRE within a data mining system on a cloud infrastructure.

The proposed cloud oriented service setup is going to work according the following principles: Input data containing a single table of entities and one or more event tables are loaded from the database and sent to the Load manager.

The Load manager splits this data into chunks, where each chunk contains a subset of entities with its related events. This subset should be small enough in order to be processed in reasonable time. However it should be also large enough in order to contain reasonable variations in data distribution. The data chunks are then serviced via the Load manager to multiple processing clusters, which contain a separate virtual container for every service (APG, ABRE or other), effectively distributing the workload across the cloud infrastructure similar to the *map reduce* principles. Rules produced in each cluster are finally sent to the Rule aggregator which assembles final prediction model and sends it either back to the database or to other cloud service which may have requested it.

III. Implementation Remarks

The APG method is implemented in Python using NumPy and SciPy packages in order to calculate the correlation coefficients between processed attributes. The aggregations generator is of our own implementation as well as the HAC filter/eliminator. The service messaging interface is established using the Flask web framework. The service can process either CSV files or JSON messages. However, the communication protocol of the APG is established mainly using JSON. The message is composed of the header, which contains the information about required processing task such as the target attribute and also the meta-information such as the table keys, IDs, attribute safe list etc. The rest of the message contains entity table and event table(s).

The ABRE method is currently implemented as a process in Rapid Miner Studio. It is able to process the CSV files only. Prospectively it is planned to export it as a web service using Rapid Miner Server and set up its messaging interface in order to be directly compatible with APG and other planned cloud services (using JSON messages).

Both services will be deployed using the Docker container technology in quasi-separate virtual environments. The Python container for APG is already finished and preliminary tested. The Rapid Miner Server container is also completed and tested, however the ABRE service implementation is due to be done.

IV. Preliminary Results and Discussion

As our cloud infrastructure [3] is still in the stage of preparation we were not yet able to implement and test the proposed system as a whole. However we can offer some preliminary results of the individual modules. The testing database contains historical data about 10 002 customers (i.e. entities) and also 10 002 related purchase events. However, 2541 customers are missing the target attribute; therefore only 7461 are applicable for further analysis. The customer table contains 2 applicable entity attributes, one of which is selected as the target attribute. The events table contains 8 applicable attributes with a total of 80 016 values. Since 2 attributes are nominal and 6 numeric, the APG algorithm can produce 2*4+6*6=44 unique aggregated attributes. These attributes are further filtered to retain only the pre-specified amount of 3 (best) predictors. Whole APG processing is done single threaded on a 2 GHz core approximately in 5 seconds. The ABRE has used these predictors and the remaining single entity attribute in order to create 17 rules in approx. 3 seconds (also on a single 2 GHz core). The most significant rule covers 6581 examples and classifies correctly 3600 of them. The second most significant rule covers 496 examples and classifies 322 of them correctly.

The preliminary results confirm the general functionality of the methods, especially performance-wise. However, in order to obtain better results of the analysis, it is necessary to gear more effort towards proper setting of decision tree generation parameters. We would also like to test the system on more datasets from various application domains. Our current main goal is the completion of the service interface implementation and the deployment of the system on the projected cloud platform. Afterwards we look forward to conclude the setting of the system parameters and present more detailed results.

Acknowledgement

This paper is prepared within the Project implementation: University Science Park TECHNICOM for Innovation Applications Supported by Knowledge Technology, ITMS: 26220220182, supported by the Research & Development Operational Programme funded by the ERDF. "We support research activities in Slovakia/this project is being co-financed by the European Union"(50%). This paper is also supported by KEGA, Grant Agency of Ministry of Education of Slovak Republic, under *Grant No. 014TUKE-4/2015* – "Digitalization, virtualization and testing of a small turbojet engine and its elements using stands for modern applied lecturing"(50%).

References

[1] H. Hu, Y. Wen, T.-S. Chua, X. Li, "Toward Scalable Systems for Big Data Analytics: A Technology Tutorial," *Access, IEEE* , vol. 2, pp.652-687, 2014, doi: 10.1109/ACCESS.2014.2332453

[2] R. Branch et al., "Cloud Computing and Big Data: A Review of Current Service Models and Hardware Perspectives," *Journal of Software Engineering and Applications*, vol.7, no.8, pp. 686-693, 2014, DOI:10.4236/jsea.2014.78063

[3] J. Paralič et al., "IT tools and services for analyses of different types of processes," in *University Science Park TECHNICOM*, Košice, 2014, pp. 16, ISBN 979-80-8086-240-4

[4] J. Paralič et al., "IT tools and services for analyses of different types of processes," in *Modely fungovania vedeckých parkov a výskumných centier : skúsenosti a príležitosti pre Slovensko*, Košice, 2015, pp. 53, ISBN 978-80-8086-246-6

[5] J. Paralič, J. Kováč, "Automatic generation of predictors from semi-structured data - Project task definition", unpublished, May 2014.

[6] L. Rokach, O. Maimon, "Decision trees." In *Data Mining and Knowledge Discovery Handbook*, Springer US, 2005, pp. 165-192. ISBN 978-0-387-25465-4

[7] J. R. Quinlan, "Generating Production Rules from Decision Trees."in IJCAI. vol. 87. 1987.

[8] M. L. Zepeda-Mendoza, O. Resendis-Antonio, "Hierarchical Agglomerative Clustering," *Encyclopedia of Systems Biology*, Springer New York, 2013, pp. 886-887, ISBN 978-1-4419-9863-7

[9] D. Müllner, "Modern hierarchical, agglomerative clustering algorithms." arXiv preprint arXiv:1109.2378, 2011.

[10] J. H. Friedman, "A Recursive Partitioning Decision Rule for Nonparametric Classification", *IEEE Transactions on Computers*, vol.26, no. 4, pp. 404-408, April 1977, doi:10.1109/TC.1977.1674849

[11] R. I.Lerman, S. Yitzhaki, "A Note on the Calculation and Interpretation of the Gini Index," *Economics Letters*, vol. 15, no. 3-4, pp. 363–368, 1984, doi:10.1016/0165-1765(84)90126-5