VŠB–TU Ostrava, FEECS, Department of Computer Science
Charles University in Prague, MFF, Department of Software Engineering
Czech Technical University in Prague, FEE, Department of Computer Science
Czech Society for Cybernetics and Informatics

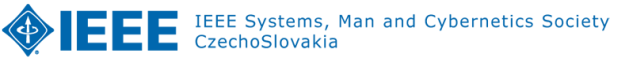
Proceedings of the Dateso 2013 Workshop

DATESO 2013

Databases, Texts, Specifications, and Objects

http://www.cs.vsb.cz/dateso/2013/
http://www.ceur-ws.org/Vol-971/

Supported by



http://www.mirlabs.org/ http://arg.vsb.cz/ieee-smc/

April 17 – 19, 2013
Písek

DATESO 2013

Technical editor:
Pavel Moravec, pavel.moravec@vsb.cz
VŠB – Technical University of Ostrava
Faculty of Electrical Engineering and Computer Science
Department of Computer Science

Page count:	158
Impression:	150
Edition:	1^{st}
First published:	2013

This proceedings was typeset by PDFLaTeX.
Cover design by Pavel Moravec (pavel.moravec@vsb.cz) and Tomáš Skopal.
Printed and bound in Ostrava, Czech Republic by TiskServis Jiří Pustina.

Published by VŠB – Technical University of Ostrava
FEECS, Department of Computer Science
17. listopadu 15, 708 33 Ostrava-Poruba, Czech Republic

Steering Committee

Václav Snášel	VŠB-Technical University of Ostrava, Ostrava
Karel Richta	Czech Technical University, Prague
Jaroslav Pokorný	Charles University, Prague

Program Committee

Václav Snášel (chair)	VŠB-Technical University of Ostrava, Ostrava
Jaroslav Pokorný	Charles University, Prague
Karel Richta	Czech Technical University, Prague
Peter Vojtáš	Charles University, Prague
Michal Krátký	VŠB-Technical University of Ostrava, Ostrava
Tomáš Skopal	Charles University, Prague
Pavel Moravec	VŠB-Technical University of Ostrava, Ostrava
Irena Mlynková	Charles University, Prague
Michal Valenta	Czech Technical University, Prague
Pavel Loupal	Czech Technical University, Prague
Martin Nečaský	Charles University, Prague
Jiří Dvorský	VŠB-Technical University of Ostrava, Ostrava
Radim Bača	VŠB-Technical University of Ostrava, Ostrava
Tomáš Knap	Charles University, Prague
Pavel Strnad	Czech Technical University, Prague
Ondřej Macek	Czech Technical University, Prague

Organizing Committee

Pavel Moravec	VŠB-Technical University of Ostrava, Ostrava
Yveta Geletičová	VŠB-Technical University of Ostrava, Ostrava

Preface

DATESO 2013, the international workshop on current trends on Databases, Information Retrieval, Algebraic Specification and Object Oriented Programming, was held on April 17 – 19, 2013 in Písek.

The 13th year was organized by Department of Computer Science VŠB-Technical University Ostrava, Department of Software Engineering MFF UK Praha, Department of Computer Science and Engineering FEL ČVUT Praha, and Working group on Computer Science and Society of Czech Society for Cybernetics and Informatics. The DATESO workshops aim for strengthening connections between these various areas of informatics, particularly this year, Sematic Web, semistructured data, social networks, and formal specifications.

The proceedings of DATESO 2013 are also available at DATESO Web site: `http://www.cs.vsb.cz/dateso/2013/` and CEUR Workshop Proceeding site: `http://www.ceur-ws.org/Vol-971/` (ISSN 1613-0073). The Program Committee selected 14 papers (7 full papers and 7 posters) from 21 submissions, based on two independent reviews.

We wish to express our sincere thanks to all the authors who submitted papers, the members of the Program Committee, who reviewed them on the basis of originality, technical quality, and presentation. We are also thankful to the Organizing Committee and Amphora Research Group (ARG, `http://www.cs.vsb.cz/arg/`) for preparation of workshop proceedings. Special thanks belong to Czech Society for Cybernetics and Informatics

April, 2013 V. Snášel, K. Richta, J. Pokorný (Eds.)

Table of Contents

Full Papers

On Estimation of the Spatial Clustering: Case Study of Epidemiological Data In Olomouc Region, Czech Republic 1
Lukáš Marek, Vít Pászto, Jiří Dvorský, Pavel Tuček

Towards a Runtime Code Update in Java 13
Marcel Hlopko, Jan Kurš, Jan Vraný

Are Shape Metrics Useful for a Geocomputation? CORINE Land-Cover Analysis Case Study 26
Vít Pászto, Lukáš Marek, Pavel Tuček

QuickXDB: A Prototype of a Native XML DBMS 36
Petr Lukáš, Radim Bača, Michal Krátký

Efficient in-memory data structures for n-grams indexing 48
Daniel Robenek, Jan Platoš, Václav Snášel

P system based model of passenger flow in public transportation systems: a case study of Prague Metro 59
Zbyněk Janoška, Jiří Dvorský

How can formalization of SOA help in finding solutions for IT systems . . . 70
Zdeněk Skřivánek, Karel Richta

Short papers

Comparative Summarization via Latent Dirichlet Allocation 80
Michal Campr, Karel Jezek

Using Retinex and SVD Algorithms for Detection of Frayed Edge in Steel Plate 87
Michal Holiš, Martin Plaček, Jiří Dvorský, Jan Martinovič, Pavel Moravec

Application of Relative Derivation Terms by Polynomial Neural Networks 98
Ladislav Zjavka

Evolution of Co-Authors Communities Formed by Terms on DBLP 109
Alisa Babskova, Pavla Dráždilová, Jan Martinovič, Václav Svatoň, Václav Snášel

A Linguistic Method into Stemming of Arabic for Data Compression 119
Hussein Soori, Jan Platoš, Václav Snášel

Searching Time Series Based On Pattern Extraction Using Dynamic Time Warping 129
Tomáš Kocyan, Jan Martinovič, Pavla Dráždilová, Kateřina Slaninová

On Updating in XML Peer-to-Peer Databases 139
Adam Šenk, Michal Valenta

Author Index 149

On Estimation of the Spatial Clustering: Case Study of Epidemiological Data In Olomouc Region, Czech Republic

Lukáš Marek, Vít Pászto, Jiří Dvorský, Pavel Tuček

Department of Geoinformatics, Faculty of Science, Palacky University in Olomouc
17. listopadu 50, Olomouc, 771 46, Czech Republic
{lukas.marek, pavel.tucek}@upol.cz, vit.paszto@gmail.com,
jiri.dvorsky@vsb.cz

Abstract. An evaluation of spatial patterns and a clustering play an important role among methods of spatial statistics. However, traditional clustering techniques are seldom suitable for analyses of spatial data and patterns because they usually do not count on spatial relations and qualities of objects. This paper aims to introduce usage of methods of spatial clustering estimation, which are based mainly on the position of events and not only on the events attribute space. Firstly, the methods of the spatial clustering or randomness estimation are introduced and applied on a real dataset, then spatial clusters are identified and the intensity of processes is quantified. Non-spatial properties and a time are considered together with the location data. Also methods of the multivariate statistics are used for the purpose of the classification of regions with similar properties. Particularly, occurrence data of selected infectious diseases in Olomouc Region in period 2004 – 2010 provided by Regional Public Health Service in Olomouc are used for the case study.

1 Introduction

An application of the geographical information system, as well as a spatial statistics, for the exploration of the spatial pattern of health data has been highly discussed in the literature [2, 14, 18] and it is one of top topics in geosciences nowadays [5]. The literature also often uses phrases geographical epidemiology, spatial epidemiology, medical geography or even geomedicine that describe dynamic body of the theory and analytic methods concerned with the study of spatial patterns of the disease incidence and mortality [22]. Since John Snow's famous geographical study of the cholera in London in 1854 century, the interconnection of health data and analyses of spatial patterns became a standard procedure. Possibilities of current geographical information systems together with properties of databases, where health and epidemiological data are stored, make spatial evaluation easier than anytime before [13].

This contribution presents the usage of methods of spatial statistics applied on the epidemiological data from Olomouc region – one of administrative units of Czech

© V. Snášel, K. Richta, J. Pokorný (Eds.): Dateso 2013, pp. 1–12, ISBN 978-80-248-2968-5.
VŠB – Technical University of Ostrava, FEECS, Department of Computer Science, 2013.

Republic corresponding with NUTS3. Cases of one particular infectious disease – parotitis - are assessed within the period 2004-2010. Records about infection come from the database of infectious diseases, which is called EPIDAT. This database contains information about every single case of the infection which is reported by local doctors. EPIDAT also contains data about infected patients as the place of residence, the place of infecting, the time of treatment, the way of isolation etc. Although exact addresses of patients are reported, due to the necessity of preserving anonymity only approximate addresses are provided. That is why the exact geocoding is unfeasible and all records are aggregated into the regular fishnet or randomized within it.

2 Case study and Data

This section intents to describe data and their adjustment for the usage within techniques of Exploratory Spatial Data Analysis (ESDA) and Local Indicators of Spatial Association. Epidemiological data comes from the EPIDAT (EPIdemiological DATabase), which stores mandatory records about all infectious diseases in the Czechia. The case study is dealing mainly with spatial attributes of one selected disease – parotitis (i.e. mumps). Firstly, geocoded data are visualized in the form of (false) choropleth maps. Choropleth maps [12] allows to the researcher first, so called, "visual" analysis of spatial structure. Then quantitative analyses of possibilities of spatial clustering are proceeding – a kernel density estimation and G-function [3]. At last, data are aggregated and randomized into the regular hexagonal fishnet and spatial autocorrelation in local scale is explored using Moran's I, Getis Ord General G and LISA, which are comprehensively described in [1, 10]. Software tools used for the realization of case study was ArcGIS (choropleth maps, LISA) and R-project with packages for the spatial statistics, manipulating with spatial data and the visualization (spatstat, maptools, spdep, etc.).

The analyzed disease is a parotitis. Parotitis is an inflammation of one or both parotid glands, the major salivary glands located on either side of the face, in humans. The parotid gland is the salivary gland most commonly affected by an inflammation. The Mumps is an airborne virus and can be spread by an infected person coughing or sneezing and releasing tiny droplets of contaminated saliva, an infected person touching their nose or mouth, then transferring the virus onto an object, or sharing utensils [17]. Routine vaccinations have dropped the incidence of mumps to a very low level [15].

2.1 Data

The EPIDAT database is used to ensure the mandatory reporting, recording and analysis of infectious diseases in the Czech Republic. The database is used nation-widely by Public Health Service of the Czech Republic from 1st January 1993. The reporting of infectious diseases is the legal basis for local, regional, national and international control of infectious diseases (EU, WHO). The data storage is used to

secure exchange of actual data sets on the prevalence of infections among the departments of Public Health Service of CR, Ministry of Health of CR and Public Health Institute in Prague.

Total of 53 diagnoses of infectious diseases are monitored into the EPIDAT database. Each record contains 50 attributes. In terms of spatial analyses, the most important properties are information about the patient's residence as well as the place of infection and the place of sicken (the place where the patient became ill, often place of clinic or doctor's office).

The data set for this study was provided by the Regional Public Health Service in Olomouc. The original provided dataset contains 32 698 records of 11 selected infections from 53 diagnoses and covers the period 2004-2010, but only 958 records of one selected disease – parotitis. Because it is treated with sensitive personal data, the name, surname, identity number and full address is not included. Furthermore, geocoded data were randomized and anonymized using aggregation from the street network and municipality membership into the regular hexagonal fishnet, which is usual procedure in spatial epidemiology and econometrics [10]. The problem of aggregating the point based spatial phenomena into the district is well known as MAUP – Modifiable Area Unit Problem.

EPIDAT database is filled with data manually and it is a transcription of medical records. That situation guarantees the occurrence of errors, mistyped characters and different kinds of used abbreviations. A manual control and subsequent correction of mistakes could be time consuming and almost impossible because of the amount of records in database. That is why a tool for semi-automatic control, repairing and geocoding was developed as the result of collaboration between Department of Geoinformatics and Regional Public Health Service in Olomouc. Owing to this tool, whole process of repairing wrong records is time acceptable. Besides using our developed tool, the Google Geocoding API is used for geocoding addresses. Google Geocoding API substitutes the physical ownership of complete street data and the database of addresses and allows to geolocate with a suitable precision [23]. The process of geocoding of 32 698 addresses took 49 469 seconds (13.75 hours).

Several requisite steps have been done before any global or local analyses were executed. Firstly the data of selected infection diseases, as well as complete data, were randomized and aggregated into the regular hexagonal fishnet. Each hexagon has the area of 6.25 km^2, which is similar to the area of the cell established by Morishita index [16] and coincidentally, it is corresponding to the area of the average cadastral unit in the region. A number of inhabitants in hexagons was estimated from cadastral units and municipalities with usage of the areal weighting. Both, absolute and relative (prevalence on 1000 population) aggregation units were created but only the absolute occurrence entered the analysis. Secondly, spatial weight matrices were generated for each input of the disease. K-nearest neighbors method, with K = 12, was selected as the way of the spatial conceptualization, the other way is an assessment of the maximum threshold (distance) of possible connections among cases. The example of the spatial weight matrix is shown in the Figure 1.

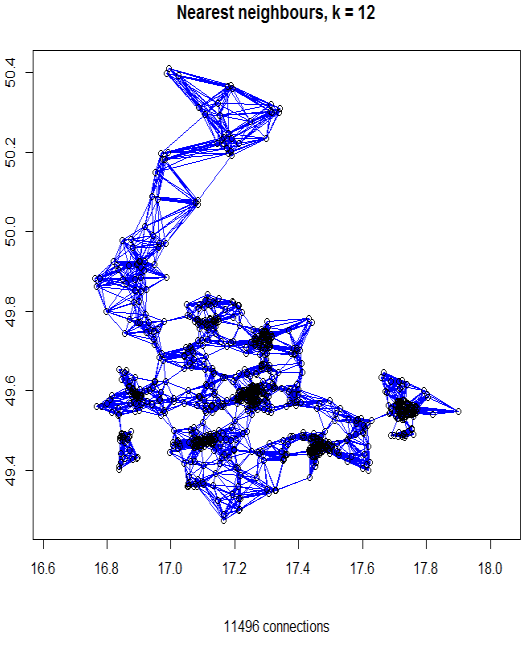


Fig. 1. The example of the generated spatial weight matrix. Points represent pseudo-randomized cases of the disease. Lines symbolize links among points, each case is connected to 12 other points / neighbors without taking into consideration of the distance

3 Methods

Analyses of spatial pattern of diseases occurrence, as well as their relations to potentially risk factors of the environment, are important parts of health studies. According to [7] three main broad areas of spatial epidemiology can be identified:

- disease mapping,
- geographic correlation studies,
- clustering, disease clusters, and surveillance.

The presented study is mainly focused on methods of the estimation and assessment of a spatial clustering as well as a multivariate clustering. Firstly, choropleth maps were constructed. Subsequently, the methods of the spatial clustering or randomness estimation were applied on the dataset and then spatial clusters are identified and the intensity of processes is quantified. Finally, the multivariate statistical clustering is executed, which extends the previously proceeded analyses.

Methods, which are introduced and described later in this chapter, are often covered by a common name Exploratory Data Analysis (EDA). This label

incorporates a wide group of statistical techniques that are very useful for both, an initial and deeper exploration of patterns and relations within the given data structure. In case of involving spatial metrics, the group of methods is called Exploratory Spatial Data Analysis (ESDA). The list of methods presented below in the text is not a complete enumeration, but only a brief overview of elementary techniques applicable to the problems and questions connected to the geographical space. It is worth to note, that most of E(S)DA methods are traditional techniques with basis in last century, but the progress in the (geographical) information science allows their wider usage.

3.1 Choropleth maps

Choropleth maps are probably the most common type of map for the display of areal data. These maps use different color and pattern combinations to depict different values of the attribute variable associated with each area, which is colored according to the category to which its corresponding attribute value belongs [22]. Although choropleth maps do not show continuously distributed values, they often portray densities [19]. Viewed in this way, one can consider them as a primitive visual tool for the analysis of spatial distribution of phenomenon.

3.2 Identification of Spatial Processes

A huge amount of methods for the estimation of the prevalent type of processes in the area are based on the testing of Complete Spatial Randomness (CSR) [6], the visual comparison of the plotted function with CSR or quadrat counts. E.g. quadrat test, G-function, K-function or Morishita index and Fry plot belong among these methods. Other suitable method is then a density kernel. It is appropriate to mention that the identification of spatial processes with the usage of previously presented methods is based mainly on the location of disease cases.

The evaluation of the spatial pattern of parotitis in this paper is realized by the plot of Morishita index and assesment of G-function. Morishita defined an index of spatial aggregation for a spatial point pattern based on quadrat counts. The spatial domain of the point pattern is first divided into Q subsets (quadrats) of equal size and shape [16]. The number of points falling in each quadrat is counted. If the pattern is completely random, index should be approximately equal to 1; values greater than 1 suggest clustering. Morishita plot is also helpful with an assessment of distances within clusters and also with the estimation of the pixel size or aggregating units in case of anonymization, for which the method of searching for the break point of the biggest change, so-called “elbow” method, is used.

G-function is the nearest neighbour distance distribution function (i.e. empirical cumulative distribution function of nearest neighbours). The shape of plotted G function is usually compared with the simulated envelope of random processes. This comparison allows unhiding the type of the possible spatial pattern. If the curve of G function takes place above the CSR envelope, then clustering is assumed. Its position

below the envelope means regular patterns and the position within the envelope refers to the random pattern [3].

3.3 Global and Local Spatial Clustering

Spatial autocorrelation is the correlation among values of a single variable strictly attributable to their relatively close locations on a two-dimensional (2-D) surface, introducing a deviation from the independent observations assumption of classical statistics [9]. Tobler's first law of geography encapsulates this situation, *"everything is related to everything else, but near things are more related than distant things".* The positive spatial autocorrelation refers to patterns where nearby or neighboring values are more alike; while the negative spatial autocorrelation refers to the patterns where nearby or neighboring values are dissimilar. One can distinguish two main types of spatial autocorrelation, which are global and local autocorrelation.

These techniques are collectively denoted as Exploratory Spatial Data Analysis (ESDA) and Local Indicators of Spatial Association (LISA), which are widely spread in geosciences and GIS software. Comprehensive description of theory, as well as detail examples of usage, can provide e.g. [1] or [10].

3.4 Multivariate Clustering

While the spatial clustering creates groups, which are based mainly on the similar location or the location and one common characteristic, methods of the multivariate statistics deals with an inverse situation. Thus, an aim of multivariate clustering is to categorize set of object with the emphasis on their quantitative and/or qualitative characteristics but mostly without implementing spatial dependencies [20], albeit several attempts for the combination of both approaches have appeared in recent years [4, 11].

For the purpose of this study, several methods of multivariate clustering, which were evaluated as the most suitable by simulations, were performed. Firstly, the similarity among all cases of parotitis through time is evaluated using the hierarchical clustering method with average linkage based on the Sokal-Michener dissimilarity distance measure for nominal variables [21]. Then similarity among hexagonal areas with aggregated values is calculated using Partitioning Around Medoids (pam) clustering algorithm based on the squared Euclidean dissimilarity distance measure. Moreover, DBSCAN algorithm - A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise [8] – enables a different approach to the estimation of clusters, which is based on the internal density of clusters and is highly suitable for the usage with spatial database. The algorithm provides results, which are visually similar to the density kernel in case of incorporating simple locations. But it is more efficient in minimal requirements of domain knowledge to determine the input parameters, discovery of clusters with arbitrary shape and good efficiency on large databases [8].

4 Results

Two choropleth maps are constructed for the purpose of case study (Fig. 2). The first map (*left part*) expresses occurrences (i.e. absolute number) of parotitis in the municipalities of Olomouc region between January 2004 and December 2010. Darker areas mean that higher absolute number of cases was reported from the area. The second map expresses relative measure – prevalence, i.e. the number of cases of parotitis in the population of municipality. Both maps show that the more populated southern part of region is more affected by the parotitis than the northern part because the darkest areas in the first map match the biggest towns in the region. But second map is more particular and allows specifying of several centres.

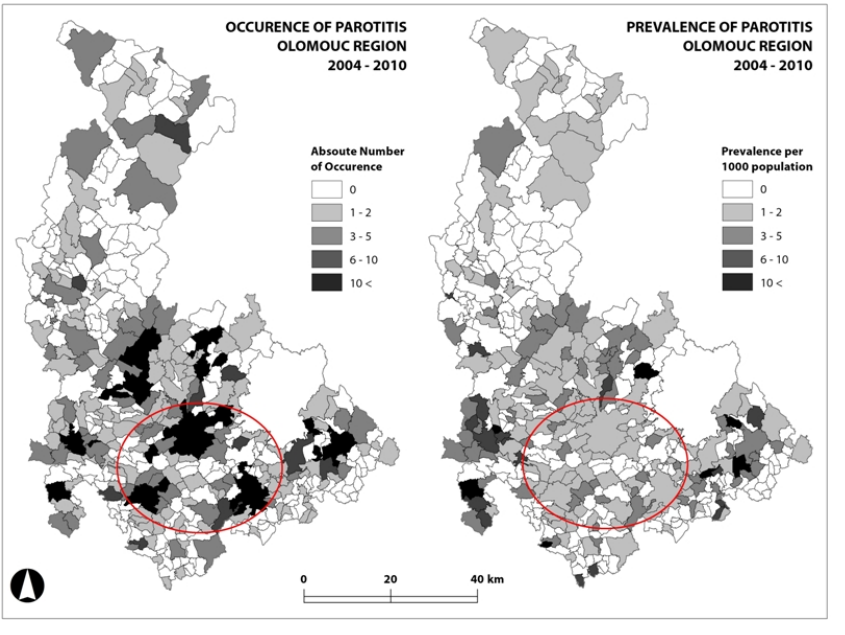


Fig. 2. Visualization of absolute (occurence - *left*) and relative (prevalence - *right*) values. Ellipses indicate places with high probability of misinterpretation

Results of Morishita index plot and the comparison between G-function and a simulated CSR process (Fig. 3) prove previously predicted fact that a possibility of clustering exists in several scales in the area. With the knowledge introduced before, it is evident that according to MI plot (Fig. 3 *left part*) clustering processes dominate in the area because the progress of function descends rapidly in the first part and after the "elbow point" it starts to converge to 1. G function (Fig. 3 *right part*) is compared with the envelope of CSR after 1000 simulations. The full line expresses observed value of function G for data pattern, dotted line stands for simulated CSR and light grey regions is 96% envelope of CSR. The curve of G function appears above the CSR line up to the value 0.05, which points out to clustering processes again.

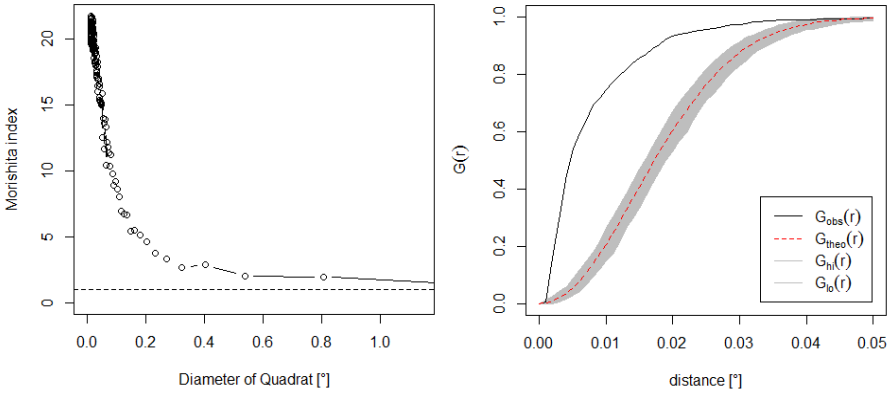


Fig. 3. Morishita index plot (*left*) and the curve G-function (*right*)

Density map is the method for quick visualization of significant clusters similar to choropleth maps or quadrat maps. The most important aspect in the estimation of kernel is not the type of kernel but its size, called bandwidth. In this study the quartic kernel with fixed bandwidth of size 0.01° (≈ 1 km) is used. This distance comes from the cross validated bandwidth selection for kernel density of point processes. Density map (Fig. 4 – *left*) then reveals primary spatial clusters, which are similar to those previously mentioned.

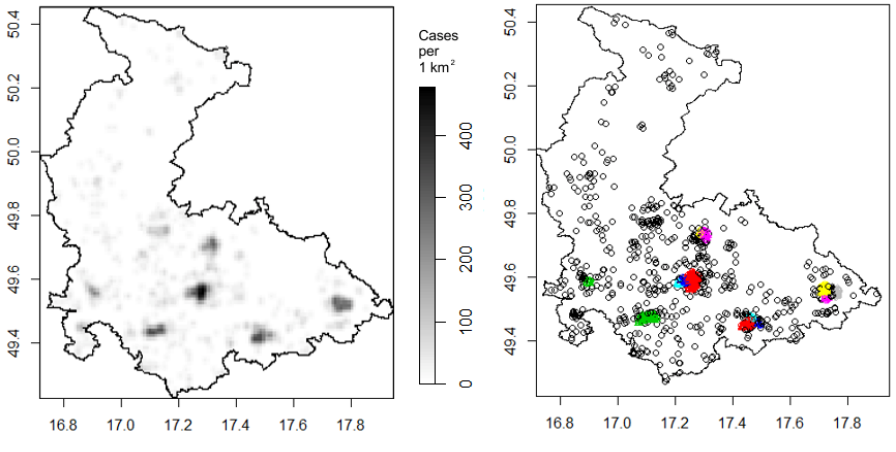


Fig. 4. Estimation of the number of cases by Kernel density (*left*) enables visual estimation of areas with dense occurrence of infection. The result of clustering by DBSCAN algorithm (*right*) –several clusters are identifiable in the southern part of study area or even in body of individual towns

Results of Moran's I, as well as Getis Ord G, prove to the presence of clustering processes. A neighbourhood for the analysis of local clustering and the size of aggregated units is based on the previous calculation of Morishita index. The global existence of clusters in the study area is proven by several methods, but their location is still not known. That is why it is proceeded to LISA.
Particular localizations of significant clusters, as well as their type, are then shown in the Figure 5. Main intensive clusters (grey areas) of high values (netted areas) with similar size of the area are identified near biggest towns in the southern and central part of the study area, while municipalities in the northern part of the region do not show any significant clusters or outliers. Clusters of high values are dominant in the area but also one outlier with low values appears.

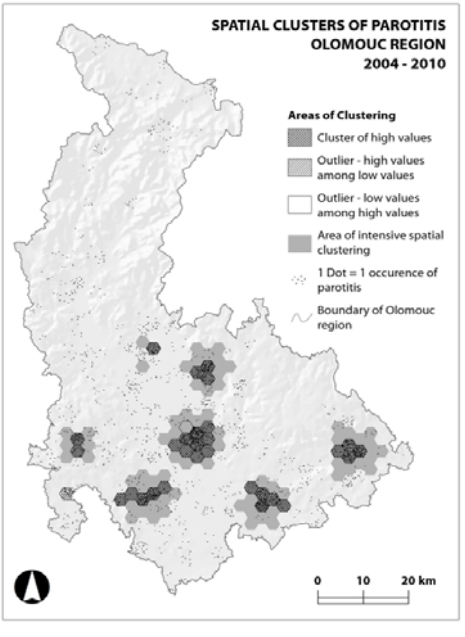


Fig. 5. Localization of disease clusters based on places of occurence of parotitis with usage of Local Indicators of Spatial Association (LISA)

For the purpose of this study, three methods of multivariate clustering, which were evaluated as the most suitable, were performed. Firstly, the similarity among all cases of parotitis through time is evaluated using the hierarchical clustering method with average linkage based on the Sokal-Michener dissimilarity distance measure for nominal variables (Fig. 6 *left part*). Then similarity among hexagonal areas with aggregated values is calculated using Partitioning Around Medoids (pam) clustering algorithm based on the squared Euclidean dissimilarity distance measure (Fig. 6 *right part*). Especially second case is useful, because it found 3 categories with a similar attribute space (categories 2-4), which corresponds with main towns in the study area and furthermore it divides these towns in separate classes.

Result of the third method is shown in the Figure 4 (*right part*). DBSCAN algorithm is highly effective for the estimation of spatial clusters even in the noise data. Fourteen clusters are found in the health data in the case study. These clusters not only correspond with settlements in the area of interest but also express inner differences in clusters and divide town into separate zones.

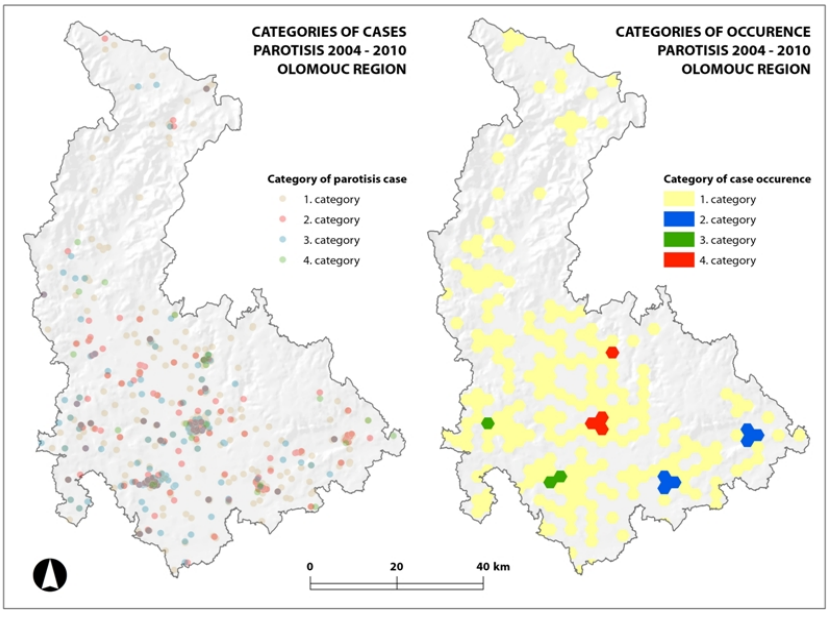


Fig. 6. Spatial visualization of multivariate clustering - similar cases (*left*), similar locations (*right*)

5 Discussion and Conclusion

One can easily explore spatial pattern and spatial relations with usage of spatial statistics and especially methods of spatial clustering estimation. But results of spatial statistics and mainly their interpretation are usually experience-dependent, i.e. subjective. Plenty of spatial techniques are also scale-dependent, thus their results are very sensitive on the precise adjustment of parameters and one small change can cause important differences in results. At least one example is given in this paper, which is an estimation of the kernel for density maps. Both, global and local indices of spatial autocorrelation are sensitive on the selection of parameters as well. The evaluation of spatial patterns is influenced also by others environmental factors, which some of them may be primarily hidden. Demographic factors are mostly the most influential element in case of health datasets.

Earlier presented procedures and techniques are only sample of suitable tools for spatial statistics, their further development and implementation lead not only to

spatial but to space-time and space-time-attribute analyses. Which are very complex and I believe they will be the predominate type of analyses in near future.

This paper presented techniques of the exploration of the spatial pattern. The usability of methods has been proved on the case study. An exploration of spatial patterns of the occurrence of parotitis (mumps) in the Olomouc Region was chosen as the model situation. Methods of EDA and ESDA confirmed the hypothesis that clustering processes exist in the area. Firstly the randomness of occurrence was tested and then the predominant type of the process was searched with a help of visualizing methods (choropleth maps and density maps) and tested (Morishita index plot). The spatial autocorrelation was explored on the aggregated data in the form of regular hexagons. Several clusters have been identified using methods of LISA. This clusters and their description are depicted on the map (Fig. 4). Clusters of high values with intensive processes are prevailing in around the towns Olomouc, Hranice, Přerov, Prostějov, Šternberk a Konice. Spatial statistics allows outstanding possibilities of exploration of spatial and space-time patterns, although some of methods have their strict limits and their interpretation can be subjective and experience dependent.

At last, data were evaluated by methods of multivariate statistics, which served to the searching of similar cases and regions with similar characteristics of disease occurrence.

Acknowledgement

The authors gratefully acknowledge the support by the Operational Program Education for Competitiveness - European Social Fund (project CZ.1.07/2.3.00/20.0170 of the Ministry of Education, Youth and Sports of the Czech Republic).

References

1. Anselin, L.: Local indicators of spatial association—LISA. Geographical analysis. 27, 2, (1995).
2. Bergquist, R.: New tools for epidemiology: a space odyssey. Memórias do Instituto Oswaldo Cruz. 106, 7, 892–900 (2011).
3. Bivand, R.S. et al.: Applied Spatial Data Analysis with R. Springer New York, New York, NY (2008).
4. Carvalho, A. et al.: Spatial Hierarchical clustering. Rev. Bras. Biom. 27, 3, 411–442 (2009).
5. Davenhall, B.: Geomedicine: Geography and Personal Health. Esri, Redlands (2012).
6. Dixon, P.M.: Ripley 's K function. In: El-Shaarawi, A.H. and Piegorsch, W.W. (eds.) Encyclopedia of Environmetrics. pp. 1796–1803 (2002).
7. Elliott, P., Wartenberg, D.: Spatial Epidemiology: Current Approaches and Future Challenges. Environmental Health Perspectives. 112, 9, 998–1006 (2004).
8. Ester, M. et al.: A density-based algorithm for discovering clusters in large spatial databases with noise. Proceedings of the 2nd International …. (1996).

9. Griffith, D., Arbia, G.: Detecting negative spatial autocorrelation in georeferenced random variables. International Journal of Geographical Information Science. 24, 3, 417–437 (2010).
10. Haining, R.: Spatial Data Analysis: Theory and Practice. Cambridge University Press (2004).
11. Horák, J. et al.: Methods of Spatial Clustering in a City. Geografie a Geoinformatika - Výzva pro praxi a vzdělávání. pp. 1–11 (2011).
12. Koch, T.: Cartographies of Disease: Maps, Mapping and Medicine. ESRI Press, Redlands, CA (2005).
13. Marek, L. et al.: Spatial Analyses of Epidemiological Data: Case Study In Olomouc Region. 12th International Multidisciplinary Scientific GeoConference SGEM: SGEM 2012, Proceedings Volume II. pp. 1155 – 1162 STEF92 Technology Ltd, Sofia, Bulgaria (2012).
14. Meade, M.S., Emch, M.: Medical geography. The Guilford Press, New York, NY (2010).
15. Medscape-Reference: Parotitis, http://emedicine.medscape.com/article/882461-overview.
16. Morishita, M.: Measuring of the dispersion of individuals and analysis of the distributional patterns. Memoir of the Faculty of Science. pp. 215 – 235 Kyushu University (1959).
17. NHS, Choices: Mumps - Causes, http://www.nhs.uk/Conditions/Mumps/Pages/Causes.aspx.
18. Ricketts, T.C.: Geographic information systems and public health. Annual review of public health. 24, 1–6 (2003).
19. Rushton, G.: Public health, GIS, and spatial analytic tools. Annual review of public health. 24, 43–56 (2003).
20. Tabachnick, B., Fidell, L.: Using multivariate statistics. Pearson (2007).
21. Walesiak, M., Dudek, A.: Symulacyjna optymalizacja wyboru procedury klasyfikacyjnej dla danego typu danych – charakterystyka problemu. Zeszyty Naukowe Uniwersytetu Szczecinskiego. 450, 635–646 (2007).
22. Waller, L.A., Gotway, C.A.: Applied Spatial Statistics for Public Health Data. John Wiley & Sons (2004).
23. Batch geocoder, http://mapsapi.googlepages.com/batchgeo.htm.

Towards a Runtime Code Update in Java

an exploration using STX:LIBJAVA

Marcel Hlopko[1], Jan Kurš[2], and Jan Vraný[1]

[1] Faculty of Information Technology,
Czech Technical University in Prague
{marcel.hlopko, jan.vrany}@fit.cvut.cz
[2] Software Composition Group,
University of Bern
kurs@iam.unibe.ch

Abstract. Runtime Code Update is a technique to update a program while it is running. Such a feature is often used so the developer can modify an application without the necessity to restart the application and recover desired state after restart. This saves time and lowers costs. Furthermore, there are applications which cannot be stopped, such as air traffic control systems or telephone switches. Current virtual machines for Java programming language do not support non-trivial updates to the running code. We have modified STX:LIBJAVA – an implementation of Java virtual machine within Smalltalk/X – to support arbitrary changes to the running code. Beside changes to the fields and methods which are already supported by the tools such as JRebel or Javaleon, we also support unrestricted changes to the class and interface hierarchy. Our Runtime Code Updates scheme has been integrated into the Smalltalk/X IDE, thus providing interactive environment where a developer can modify a Java application while it is running.

1 Introduction

The ability to dynamically update the code of a running application is interesting for many domains. It can reduce downtime of long-running systems by eliminating the need for stopping, redeploying and starting the application again. There are applications that have to be maintained and improved but cannot be stopped. Financial transaction processors, telephone switches, air traffic control systems, are all examples of such applications.

Dynamic code updates can improve programmer productivity during programming by giving instant feedback without the need to wait for rebuild and deployment (Shan [12]). Kabanov and Vene [7] show that many of their clients have applications which take more than 15 minutes to rebuild. It is clear that support for dynamic code updates saves a lot of development time and reduces total cost of the software product. Furthermore, runtime code update can improve debugging efficiency by not forcing the programmer to restart the program and to recreate bug preconditions.

Updating the code of the running program has been researched in the past (Fabry [3]) and is still investigated today (Dmitriev [1], Kabanov and Vene [7], Orso et al.

© V. Snášel, K. Richta, J. Pokorný (Eds.): Dateso 2013, pp. 13–25, ISBN 978-80-248-2968-5.
VŠB – Technical University of Ostrava, FEECS, Department of Computer Science, 2013.

[10], Redmond and Cahill [11], Subramanian et al. [13], Würthinger et al. [14]). Support for runtime code updates is common in VMs for dynamic languages, but not so common in VMs for statically typed languages such as Java (Ebraert and Vandewoude [2]). For example HotSpot VM[3]– the reference VM for Java– has only limited support for runtime code update. Currently, only changes to the method bodies are allowed.

There are approaches for some types of runtime code updates for Java including: JRebel[4] and Javaleon[5] – an application-level systems; Dynamic Code Evolution VM[6] – a modification of the HotSpot VM allowing runtime code changes; JVolve[7] – a solution based on the Jikes Research VM.

None of existing solutions supports all types of runtime code updates. HotSpot VM has not been developed with runtime code updates in mind and has to be modified to support this feature. Such modification requires a large amount of engineering work.

In this paper we present STX:LIBJAVA – a Java VM implementation for Smalltalk/X VM – which has been modified to support all types of runtime code updates for Java. We show solutions and implementation details which relate to the runtime code updates which may be relevant to all Java virtual machines.

The contributions of this paper are (i) presentation of the system supporting all types of runtime code updates for Java, (ii) identification of problems related to runtime code update support in STX:LIBJAVA and (iii) description of solutions to runtime code update problems in STX:LIBJAVA.

The paper is organized as follows: Section 2 describes the types of possible runtime code updates. Section 3 gives an overview of STX:LIBJAVA, a Java VM implementation used. In Section 4 we present our solutions and important implementation details. Section 5 discusses future work. In Section 6 we present related work and Section 7 concludes the paper.

2 Problem Description

2.1 Types of Runtime Code Updates

There are multiple types of runtime code updates, some of which are already implemented in the HotSpot VM, or provided by 3rd party tools executing at the application level, such as JRebel or Javaleon. More complex changes require modification of the HotSpot VM, as shown by Dynamic Code Evolution VM (DCE VM) project (Würthinger et al. [14]). Other relevant solutions make use of non-standard or research VMs, *e.g.,* JValve (Subramanian et al. [13]). A system providing full runtime code updates should handle all types of updates in Table 1, also containing a comparison of standard HotSpot VM, DCE VM, JRebel, Javaleon and JValve.

[3] http://openjdk.java.net/groups/hotspot/
[4] http://zeroturnaround.com/software/jrebel/
[5] http://javeleon.com/index.php
[6] http://ssw.jku.at/dcevm/
[7] http://www.cs.utexas.edu/ suriya/jvolve/

Feature	HotSpot	DCE VM	JRebel	Javaleon	JValve
Changes to method Bodies	✓	✓	✓	✓	✓
Adding/removing fields	x	✓	✓	✓	✓
Adding/removing methods	x	✓	✓	✓	✓
Adding/removing constructors	x	✓	✓	✓	✓
Adding/removing classes	x	✓	✓	✓	✓
Replacing superclass	x	✓	x	✓	x
Adding/removing implemented interfaces	x	✓	x	✓	x
Custom migration of changed instances	x	x	✓	✓	✓
Custom migration of changed classes	x	x	x	x	✓

Table 1. Comparison of HotSpot, DCE VM, JRebel and Javaleon features

2.2 Update of the Method Body

Out of all code updates, update of the method body is the simplest and most often used one. The signature of the method remains the same, only the code of the method is modified, for example after fixing simple bug.

As an example of this change, consider code shown in Listing 1.1. In `TicketController`, we modify the `buyButtonClicked` method. The changed method is shown in Listing 1.2.

```
1  public class TicketController {
2
3    private TicketView view;
4    private TicketsSeller seller;
5    private TicketValidator validator;
6
7    ...
8
9    public void buyButtonClicked() {
10     Ticket ticket = view.getTicket();
11     seller.sellTicket(ticket);
12   }
13 }
```

Listing 1.1. Initial code before a method body update

```
1 public void buyButtonClicked() {
2   Ticket ticket = view.getTicket();
3   if (validator.isValid(ticket)) {
4     seller.sellTicket(ticket);
5   } else {
6     throw new RuntimeException();
7   }
8 }
```

Listing 1.2. Code of the method after the update of the method body

2.3 Binary Compatible Update

Binary compatible update does not break the compatibility of the class with any existing code. Following types of updates fall into this category: adding a field[8], adding a method, adding a contructor, adding an implemented interface[9].

Following on our example in Listing 1.1, consider adding an arbitrary method. Adding this method does not break any existing code. No class depending on the `TicketController` has to be modified. But, there is an opportunity to modify other classes in the system to use newly added method (in *e.g.,* button click handler). This way the whole system can be improved and evolved at runtime.

2.4 Binary Incompatible Update

Binary incompatible update breaks compatibility with existing code. The following types of updates fall into this category: removing a field, removing a method, changing a signature of a method, replacing a superclass, removing an implemented interface. Such a situation has to be perceived and handled by the system. The runtime system can rollback the update or apply the update and throw an exception when incompatibility causes a problem.

Imagine we change the signature of the existing method. All dependent classes will keep invoking the class with the old signature, but there is no such method present in the updated class anymore. If the system allows such update, the `NoSuchMethodError` should be raised.

2.5 Updates of the Instance Format

Adding and removing a field poses an unique problem. The layout of the object changes. There may be live instances of the updated class. After the update, the instance format expected by the class is different to the instance format on the heap. And simply adding or removing the fields can bring the instance to the unexpected state not attainable by the normal execution.

2.6 Updates of the Class and the Interface Hierarchy

Updates to the class and interface hierarchy are the most complex. The methods and fields could be added or removed by updating the hierarchy therefore runtime system must be prepared for such change. From the point of view of the type safety the type correctness of the program could be broken by updating the hierarchy.

[8] Adding a field is more complex, as elaborated in Section 2.5.

[9] Adding an interface is also more complex change, as elaborated in Section 2.6

3 STX:LIBJAVA

Java is 2nd most used language nowadays having more than 17% community share[10]. Despite huge popularity, Java still lacks a runtime and development environment offering dynamic code reloading, interactive and incremental compilation. Also these features are foundations of high programming productivity of Smalltalk developers (Shan [12]).

STX:LIBJAVA is an implementation of the Java virtual machine built into the Smalltalk/X environment. In addition to providing the infrastructure to load and execute Java code, it also integrates Java into the Smalltalk development environment including browsers, debugger and other tools.

STX:LIBJAVA aims at providing fully compatible Java VM implementation, which is also capable of full interoperability with Smalltalk and vice versa. More about the architecture and interoperability features can be read at Hlopko et al. [6].

3.1 Architecture of STX:LIBJAVA

In this section we will briefly outline STX:LIBJAVA's internal architecture.

Unlike other projects which integrate Java with other languages, STX:LIBJAVA does not use the original JVM in parallel with the host virtual machine, nor does it translate Java source code or Java bytecode to any other host language. Instead, the Smalltalk/X virtual machine is extended to support multiple bytecode sets and execute Java bytecode directly.

Java runtime classes and methods are implemented as Smalltalk `Behavior` and `Method` objects. In particular, Java methods are represented as instances of subclasses of the Smalltalk *Method* class. However, they refer to the Java bytecode instead of the Smalltalk bytecode. Execution of the Java bytecode is implemented in the virtual machine. In the same way that the Smalltalk bytecode is handled by the VM, the Java bytecode is interpreted and/or dynamically compiled to the machine code (jitted).

The main disadvantage of our approach (as opposed to having a separate original JVM execute Java bytecodes) is that the whole functionality of the Java virtual machine has to be reimplemented. This includes an extensive number of native methods, which indeed involve a lot of engineering work. However, we believe that this solution opens possibilities to a much tighter integration which would not be possible otherwise.

3.2 Reference Resolving

Among other information, Java classfile contains a **Constant Pool**, a pool of constants and references used within the class (Lindholm and Yellin [9]). Constant Pool Reference (CPR) can be of following types: ClassRef, MethodRef, InterfaceMethodRef, FieldRef and StringRef. For example, ClassRef consists only of a single string constant, which contains a Fully Qualified Domain Name (FQDN) of the referenced class (Lindholm and Yellin [9]). MethodRef consists of a class ref, which identifies the class containing the method, and the *name and type* of the method.

[10] According to http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html

Every class is compiled into separate classfile. In order for this class to be used by the application, the classfile must be loaded and class needs to be **linked**. Superclass and superinterfaces are (if not already) loaded and linked before the class is linked. Final static fields have to be initialized to their constant value, and the static initializer of the class must be called (possibly setting the values of nonfinal static fields). After that the class is installed into the class registry.

```
1 public static String getNameAndParams() {
2   return String.format(
3     "%s?useUnicode=true",
4     databaseName);
5 }
```

Listing 1.3. An example of the method requiring further resolving

References are not only resolved during linking, they may be resolved also later in the runtime. Consider the method at Listing 1.3. The method contains references to the `String` class and its `format` method (used for string interpolation), but the method is not called during static initialization of the class. These references are therefore not resolved. They will be resolved, when the first invocation of the `getNameAndParams` method occurs. This lazy resolving scheme is used by STX:LIBJAVA (and HotSpot VM, DCE VM, Jikes RVM[11], JVolve and others).

Now consider the state of the VM as shown in Figure 1. In the top left corner the source code of the currently executed method is shown – the `getPaidDate` method of the `Ticket` class. In the top right corner the bytecode of the `getPaidDate` method is shown. 3 sections follow, first showing the state of the VM before the execution of the `GETFIELD` instruction, second showing the state after the `GETFIELD` was executed. The last one will be explained in Section 4.4. In each of sections on the left the constant pool of the `Ticket` class is shown, in the middle the Java Metadata Area of the VM with `Ticket` class and its field `paidDate` are shown. On the right the Java heap is shown, currently containing only one instance of the `Ticket` class. The instance consists of the header containing various fields needed by the runtime, garbage collection, synchronization etc. These are not relevant to our problem. Instance also contains a class pointer, pointing to the Java Metadata Area, where a runtime Java class representation resides. Finally, instance contains a slot for every field it should have. In our case, the `Ticket` class only has 3 fields.

Consider the state of the VM from the Part A at Figure 1. ClassRefs and FieldRefs (and the not shown MethodRefs as well) in the constant pool contain a cache field. Initially, the field is empty, but when the reference is resolved for the first time, cache is filled. Next time a reference needs to be resolved, cached value is returned immediately, without the need to lookup the class, method or field, which greatly improves the performance.

When the execution advances, `GETFIELD` instruction is to be executed. The instance which field is to be loaded is already on the stack, pushed by previous `ALOAD_0` instruction. The reference identifying the field is stored in the constant pool at the index

[11] http://jikesrvm.org/

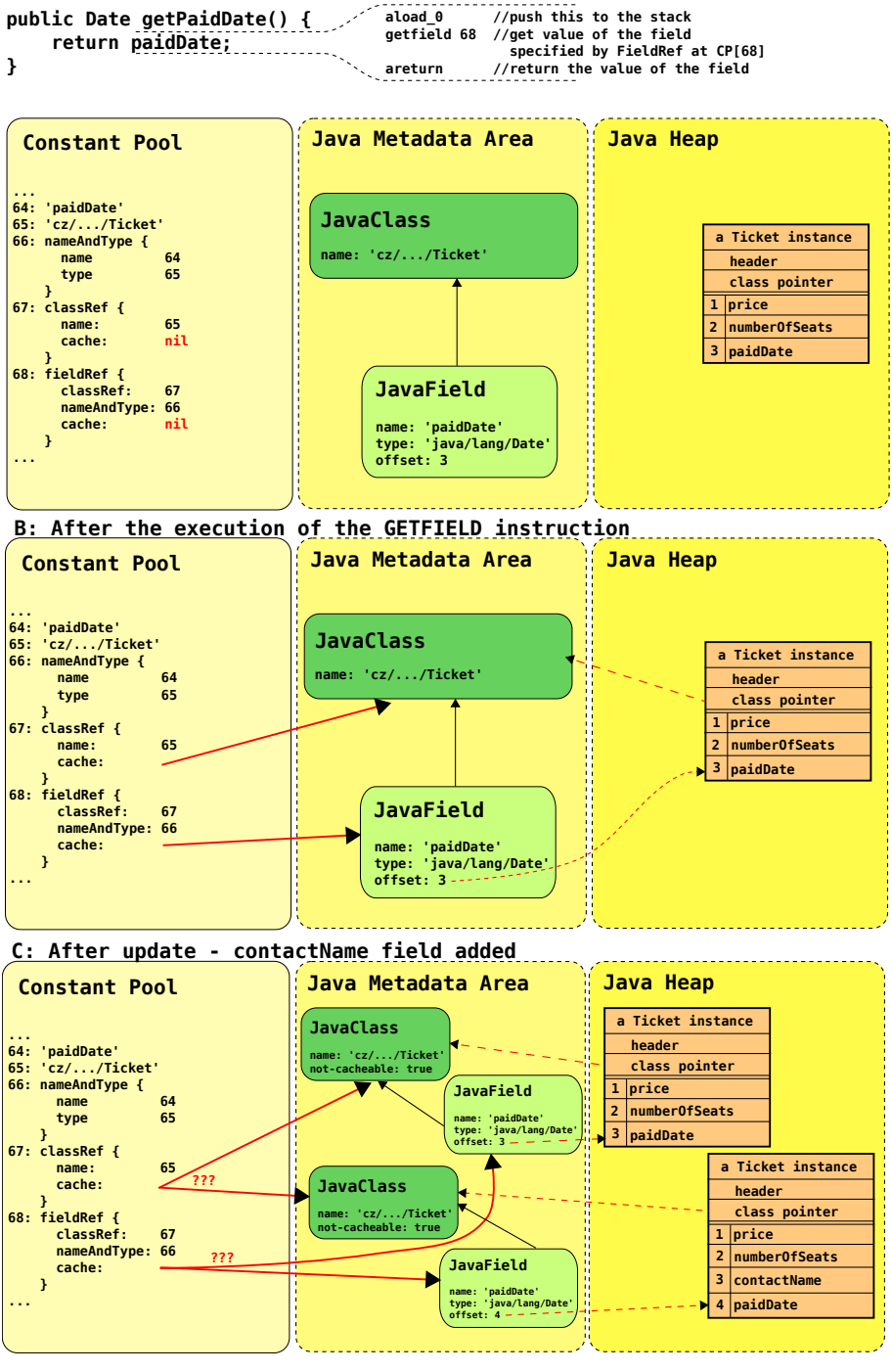


Fig. 1. State of the STX:LIBJAVA in different resolving situations

68 (68 is given as an argument to the instruction directly in the bytecode). Situation after the processing of the instruction is shown in Part B at Figure 1. A `cache` field of the ClassRef at index 67 points to the runtime representation of the `Ticket` class, `cache` of the FieldRef points to the runtime representation of the `paidDate` field of the `Ticket` class. In our situation, `Ticket` class has 3 fields, and the offset of the `paidDate` field is 3. The `GETFIELD` instruction will retrieve instance data from the heap and will jump to the 3rd slot, pushing the found value to the stack.

4 Implementation

Currently, runtime code update support in STX:LIBJAVA operates at the level of a class. Due to the absence of an incremental compiler for Java, the whole class needs to be recompiled on each update. Incremental compiler is able to compile single method, as opposed to whole Java compilation unit, as is the case of the Standard `javac` compiler – a compiler currently used by STX:LIBJAVA. When update occurs, the difference between the old and new versions of the class is found and the update is handled depending on the type.

In this section we will revisit the types of runtime code updates. For each type we will describe how STX:LIBJAVA supports the update and what were the problems affecting the implementation.

4.1 Update of the Method Body

Updates of the method bodies (described in Section 2.2) are easiest to manage. No existing code is broken and nothing on the heap has to change.

The update could cause changes in the constant pool by *e.g.,* introducing new constants or references in the method body. Therefore constant pool of the new version of the class completely replaces the old constant pool. Also, bytecodes of all methods must be replaced, as the indices into the constant pool may not be valid after the update.

In the case of Figure 1, when `getPaidDate` method changes, the constant pool of the `Ticket` class is replaced, and the bytecodes of all its methods are replaced. The change does not affect the `TicketController` or other classes which depend or use the `Ticket` class.

There may be running invocations of the updated method. These invocations are not modified and are left intact. There are attempts to replace the method as it is running (Kabanov and Vene [7], Subramanian et al. [13]) by analyzing the update and migrating the code of the method in the middle of its execution. However there are situations, when the method cannot be transparently updated (Subramanian et al. [13]). There is a large amount of engineering work involved in such an approach, and the results are not always predictable by the programmer. STX:LIBJAVA therefore does not modify running methods in favor of always accepting method body updates and predictable behavior. Therefore only new invocations are affected by the update, similarly to the behavior of the HotSpot VM (Dmitriev [1]).

4.2 Binary Compatible Update

In STX:LIBJAVA, binary compatible updates (described in Section 2.3) are handled similarly to the update of the method body. Whole constant pool and bytecodes of the methods have to be replaced. In addition, new method must be installed into the class.

Another subtle issue may arise when dealing with overloaded methods. By adding a method, an existing method could be overloaded. Java compiler is responsible to choose best-matching method based on the static types of its arguments. There may be classes in the system, which need to be recompiled and updated in order to correctly choose one of overloaded methods.

By default, STX:LIBJAVA ignores these situations. The behavior can be compared to the situation, when the application is compiled against a certain version of a particular library, and then providing different version at runtime. However, dependent classes can be recompiled when explicitly requested.

4.3 Binary Incompatible Update

The essential problem of this type of update is that it breaks existing code. An application can try to invoke non-existing methods or to store a value to non-existing field. In general, there are 2 solutions: **Allowing the update**, and throwing an exception, when removed method is accessed. This corresponds to the behavior of the HotSpot VM in the situation mentioned in Section 4.2, when a different version of a library is provided in runtime. When a method existing in compile-time but non-existing in run-time is invoked, the `NoSuchMethodError` is thrown. **Detecting the incompatibility** and not allowing the update at all (possibly rolling back other already applied updates). This approach has an advantage that the system is always in compatible state.

In STX:LIBJAVA, these updates are allowed even when they break existing code. When a legacy class tries to invoke a method, which has been removed by the update, an exception is thrown. The exception can be caught and the problem can be fixed by updating the system in runtime, *e.g.,* by implementing missing method or by updating the code calling removed method.

4.4 Update of the Instance Format

When a field is added or removed by the update, the instance format changes. In Section 3.2 we showed, that the instance on the heap contains among other unrelevant fields a single slot per declared field. After the layout changing update the fields declared by the class would be different to what currently living instances contain. Additionally, as there are resolved fields stored in the `cache` field of the FieldRef, the offset stored in resolved field may no longer be valid. Existing classes which have already cached the resolved field may access incorrect or unexisting slot, resulting in heap corruption and in abnormal application termination. Therefore this issue must be addressed by the system. There are 2 solutions – migrating the instances with update or allowing multiple versions of a class to coexist and leave old instances intact.

Updating the class and migrating existing instances is the standard approach taken by other dynamic code evolution projects for Java (Gregersen and Jørgensen [4], Kabanov and Vene [7], Subramanian et al. [13]), also due to the limitations of the HotSpot

VM and Java class loading design decisions (Liang and Bracha [8]). The main problem of this solution is how to ensure that the migrated instances are in the correct state. Solutions such as leaving added or changed fields uninitialized or initialized to the default value are not practical and can cause the instance to be in a state not achievable by normal program execution. This approach should be and usually is accompanied with the ability to specify custom migration logic (Gregersen and Jørgensen [4], Kabanov and Vene [7], Subramanian et al. [13]).

Allowing multiple versions of a single class opens the possibility to leave old instances to live with old class version, and new instances to use new class version. This is the approach taken by STX:LIBJAVA and by DCE VM (Würthinger et al. [14]). When a class is updated in such a way that the instance format changes, the old class version is removed from the Java class registry (whis is used to lookup Java classes in STX:LIBJAVA), but as the class is still referenced by the old instances it is not garbage collected until all instances die naturally. New class version is installed into the registry and all new instances are created using new class version after the installation. Advantage of the approach is that the programmer is not forced to provide migration methods for each update and still all the instances are in the valid state. In case the instances need to be migrated immediatelly with the update, the developer can explicitly provide a custom migration method and all instances will be migrated.

The migration must happen atomically, as there may be Java threads working with affected instances. The standard solution is to wait for all threads to reach and wait in a **safe-point** (Gregersen and Jørgensen [4], Kabanov and Vene [7], Subramanian et al. [13]). Thread is in a safe-point when it can yield its execution – on every method invocation, backward-jump and method return. After the migration, all threads are resumed.

Having more than one class version in runtime brings up an issue. As shown in Figure 1, FieldRefs in the constant pool, once resolved, cache the field so when the reference is accessed next time, the resolving phase can be skipped. When there are multiple versions of a single class, the offset of the field can vary depending on the class version of the current instance at runtime.

To illustrate this problem, consider Part C in Figure 1. After the update, which has added `contactName` field to the `Ticket` class, there are multiple versions of the class, as shown in Java Metadata Area. Both these versions contain `paidDate` field, but in the new version, the offset of the field is 4, on contrary to 3 in old version. To solve this problem, classes, which have multiple coexisting versions, are marked as non-cacheable. When a FieldRef is resolved, the marked classes are detected and then the resolved value is not cached. This way the resolving is performed every time.

While solving the problem, this solution is not perfect. As field access is very common operation, STX:LIBJAVA keeps the track of all old class versions in the *weak array*[12]. Everytime a reference for non-cacheable class is resolved, the weak array is checked. If the array contains only single item, it means there are no old class versions anymore, the class is marked as cacheable and resolved value can be cached again. Therefore the performance is hindered only for a limited time after the change. The concrete performance measurements have not yet been taken and are part of the future work, as mentioned in Section 5.

[12] items in the weak array are free to be collected by the garbage collector

4.5 Update of the Class and Interface Hierarchy

As described in Section 2.5, two issues must be addressed in order to handle updates to the class and interface hierarchy.

In STX:LIBJAVA, we handle these updates similarly to updates of the instance format (Section 4.4), by allowing both old and new versions of the class to coexist. In this case, the type safety of the existing instances is not broken.

Type safety can be broken, when old instances are migrated by the update, and still stored in fields of static type not type compatible with new type information. Gupta et al. [5] states that the consistency problem is undecidable. Therefore in STX:LIBJAVA it is a responsibility of the migration logic provided by the programmer to ensure that the system will be in consistent state. When an update causes an error later in the runtime, `NoSuchFieldError`, `NoSuchMethodError`, or `ClassCastException` are raised.

5 Future Work

As the runtime code update support in STX:LIBJAVA is still under development, it has not been tested in the real world yet. We expect and believe, that it would be possible to evolve a Java project from initial version (*e.g.,* first commit in the source code repository) to the latest version without the need to restart the application. Similarly, there is no performance analysis of the runtime code update support in STX:LIBJAVA done yet.

A working incremental compiler for Java would enable us to implement fully interactive environment for Java where it would be possible to start the application before it is completely finished, and to implement missing pieces in the runtime incrementally. A work on such compiler is therefore also part of our future plans.

Currently, all instances are migrated immediately, blocking the execution of all threads. While this is a solution taken by many (Gregersen and Jørgensen [4], Kabanov and Vene [7], Subramanian et al. [13]), lazy migration of instances would be beneficial for long-running applications. We are currently evaluating a solution where instances would be migrated when they are resolved.

6 Related Work

6.1 JVM HotSwap

JVM Hot Swap is a feature of Java HotSpot VM which enables, albeit limited, dynamic code updates. HotSpot VM is capable of handling changes to method bodies after implementation of the first stage of Dmitrievs (Dmitriev [1]) four-step plan in the Java platform. However, other steps have never been implemented, due to the amount of the engineering effort needed. As elaborated by Dmitriev [1], HotSpot VM has been designed for Java and with performance in mind, which makes runtime code update implementation difficult. On contrary to STX:LIBJAVA, other types of updates are not supported.

6.2 Dynamic Code Evolution VM

Dynamic Code Evolution VM (DCE VM) is a modification of the Java HotSpot VM that allows dynamic class redefinition at runtime. It allows for arbitrary code evolution changes including changes to class and interface hierarchy.

DCE VM suffers from an issue, when the dynamic type of variable does no longer match its static type, a problem described at Section 4.5 STX:LIBJAVA does not suffer from this issue and allows for all types of dynamic code changes without breaking the execution. These situations would end up raising a `NoSuchMethodError`.

6.3 JRebel, Javaleon, Javadaptor and other

There is a group of application level tools which operate on unmodified HotSpot VM, while providing runtime code update support for various types of change. These solutions make heavy use of HotSwap feature together with Java class loading. In general, some kind of proxy service is installed and this service dispatches to concrete class versions. By adding a proxy layer these solution bring certain overhead to the normal execution, even when there are no updates applied. Because proxies

JRebel is commercial tool providing support for dynamic changes including adding/removing methods and fields, but lacks the ability to dynamically change parents hierarchy (Kabanov and Vene [7]).

Javaleon is another commercial product allowing arbitrary changes to the running Java application, including changes to the parent and interface hierarchies, but updates take effect at the granularity level of components, thus Javaleon is usable only when an application is developed on top of a component system, such as NetBeans Platform or Eclipse Rich Client Platform. In order to operate, Javaleon have to preprocess the whole standard library and all 3rd party libraries used by application.

Javadaptor takes a slightly different approach. Instead of loading classes with new class loader, it performs renaming of the classes, therefore allowing them to be loaded by the existing class loader. Similarly to Javaleon and JRebel, they use bytecode manipulation and dynamic proxying in order to achieve runtime code update.

The big advantage of these tools is that they execute on the standard, unmodified HotSpot VM, allowing the runtime code updates to be used without the need to update deployment machines. However, despite large amount of engineering effort put into these projects (Kabanov and Vene [7]), they cannot be compared to customized VM feature and(or) performance-wise.

7 Conclusion

Runtime code update as a technique to update a program while it is running has proven to be valuable feature of a runtime system improving development and debugging speed while lowering the costs and downtime of long-running applications. In this paper, we presented runtime class update support in STX:LIBJAVA, a Java VM implementation for Smalltalk/X VM. To our knowledge, STX:LIBJAVA is the only publicly available virtual machine for Java that supports all types of runtime code updates. Runtime code update support enables the STX:LIBJAVA to become the first fully interactive development environment for Java.

Bibliography

[1] M. Dmitriev. Towards flexible and safe technology for runtime evolution of java language applications. In *Proceedings of the Workshop on Engineering Complex Object-Oriented Systems for Evolution*, pages 14–18. Citeseer, 2001.

[2] P. Ebraert and Y. Vandewoude. Influence of type systems on dynamic software evolution. In *the electronic proceedings of the International Conference on Software Maintenance (ICSM'05) Badapest Hungary*. Citeseer, 2005.

[3] Robert S. Fabry. How to design a system in which modules can be changed on the fly. In *Proceedings of the 2nd ICSE*, pages 470–476, 1976.

[4] A.R. Gregersen and B.N. Jørgensen. Dynamic update of java applications - balancing change flexibility vs programming transparency. *Journal of Software Maintenance and Evolution: Research and Practice*, 21(2):81–112, 2009.

[5] Deepak Gupta, Pankaj Jalote, and Gautam Barua. A formal framework for on-line software version change. *Software Engineering, IEEE Transactions on*, 22 (2):120–131, 1996.

[6] M. Hlopko, J. Kurš, J. Vranỳ, and C. Gittinger. On the integration of smalltalk and java. *Proceedings of the Intenational Workshop on Smalltalk Technologies (IWST)*, 2012.

[7] J. Kabanov and V. Vene. A thousand years of productivity: the jrebel story. *Software: Practice and Experience*, 2012.

[8] Sheng Liang and Gilad Bracha. Dynamic class loading in the java virtual machine. *SIGPLAN Not.*, 33:36–44, October 1998. ISSN 0362-1340. doi: http://doi.acm.org/10.1145/286942.286945. URL http://doi.acm.org/10.1145/286942.286945.

[9] Tim Lindholm and Frank Yellin. *Java Virtual Machine Specification, The (2nd Edition)*. Prentice Hall, Santa Clara, California 95054 U.S.A, 2 edition, 4 1999. ISBN 9780201432947. URL http://java.sun.com/docs/books/jvms/.

[10] A. Orso, A. Rao, and M.J. Harrold. A technique for dynamic updating of java software. In *Software Maintenance, 2002. Proceedings. International Conference on*, pages 649–658. IEEE, 2002.

[11] Barry Redmond and Vinny Cahill. Supporting unanticipated dynamic adaptation of application behaviour. *ECOOP 2002 - Object-Oriented Programming*, pages 29–53, 2006.

[12] Y.P. Shan. Smalltalk on the rise. *Communications of the ACM*, 38(10):102–104, 1995.

[13] Suriya Subramanian, Michael Hicks, and Kathryn S McKinley. *Dynamic Software Updates: A VM-centric Approach*, volume 44. ACM, 2009.

[14] Thomas Würthinger, Christian Wimmer, and Lukas Stadler. Dynamic code evolution for java. In *Proceedings of the 8th International Conference on the Principles and Practice of Programming in Java*, PPPJ '10, pages 10–19, New York, NY, USA, 2010. ACM. ISBN 978-1-4503-0269-2. doi: 10.1145/1852761.1852764. URL http://doi.acm.org/10.1145/1852761.1852764.

Are Shape Metrics Useful for a Geocomputation? CORINE Land-Cover Analysis Case Study

Vít Pászto[1], Lukáš Marek[1], Pavel Tuček[1,2]

[1]Department of Geoinformatics, Faculty of Science, Palacky University in Olomouc
Tř. Svobody 26, Olomouc, 771 46, Czech Republic
[2]Department of Mathematical Analysis and Applied Mathematics, Faculty of Science, Palacky University in Olomouc
17. listopadu, 771 46 Olomouc, Czech Republic
{lukas.marek, pavel.tucek}@upol.cz, vit.paszto@gmail.com

Abstract. Since shape metrics emerged in the landscape ecology as a new tool for quantitative evaluation of a landscape, it has become easier for geocomputation methods in GIS to adopt theirs principles. Nevertheless, there are still different scientific opinions about the usefulness of shape metrics. The paper describes shape metrics application for Corine Land Cover 1990, 2000 and 2006 areas (CLC) analysis along with statistical methods and discusses its benefits and disadvantages. The main goal of the paper is to evaluate CLC dataset without including attribute or qualitative information into analysis using shape metrics calculation. Thus, only geometric part of the data has been processed. Twenty eight metrics have been used for more than 900 areas (patches) from CLC dataset covering Olomouc region. Metrics values have been calculated and consequently used for correlation analysis, principal component analysis and cluster analysis. The results of the study represent complex evaluation of CLC Level 1 classes using, fundamentally, only the shape of CLC areas (patches). The analysis results show that shape metrics are very useful to identify groups of landscape patches with similar shape.

Keywords: shape metrics, GIS, land-cover, geocomputation, clustering.

1 Introduction

Since landscape ecologists can use capabilities of computer calculations, they are able to apply numerous tools to quantify landscape patches in an effective way. For this purpose, various indexes and metrics based on a patch shape have been derived, because according to [16] landscape ecology is largely founded on the notion that environmental patterns strongly influence ecological processes. Authors in [8] mentioned that developing methods to quantify landscape patterns are considered as a prerequisite to the study of pattern-process relationships. Authors in [8] continue and claim that progress has been facilitated by recent advances in computer processing and geographic information technologies.

Shape metrics are exactly those methods used for quantitative description of a patch shape, which represents real world objects. Shape and spatial metrics was

recently used in various topics, e.g. city footprint and form evaluation ([5], [14]), measuring city sprawl [15], analysis of landscape ([3], [10], [17]), in remote sensing [9] and also in a land-use change modelling [4]. Metrics are now being implemented in GIS software or extensions for GIS software but still not widely used. With the use of multivariate statistics, it is possible to evaluate, cluster and classify patches only according to their quantitative characterization. Mentioned methods are considered as a geocomputational and are both stand-alone and integrated in GIS.

There are several approaches how to classify landscape patches, but none of these are using shape metrics in combination with multivariate statistics for complex quantitative description of a landscape. It is common to use only a limited number of metrics to evaluate one specific patch group (e.g. habitats of particular species, humid areas, urbanized areas etc.). It is important to note that appropriate use of chosen metric depends on what is under the scope of study. One metric is more suitable for a one type of analysis, another for a different type. Although the use of metrics is purpose-dependent, metrics for this paper were chosen with an intention to calculate the most available ones for consequent multivariate statistics and tested if they can be (altogether) a tool for semi-automatic landscape classification. Similarly, analyzed patches used in this paper cover every patch type defined in CLC Level 1 classification nomenclature.

Thus, the approach presented in this paper is quite unique and the aim is complex landscape analysis via geocomputational methods to evaluate their usability for a landscape classification. Classification and proposed clustering methods were done with the view of the fact that only landscape patch shapes (geometry) were evaluated. Resulting clusters refer about the similarity of patch shapes and group areas with similar geometry. It is then evaluated what is the ratio of CLC Level 1 patches within clusters created only with the respect of shape metrics.

2 Data, Study Region and Methods

Analysis was performed on freely available CLC dataset from 1990, 2000 and 2006 using Level 1 nomenclature, which classifies a land cover into 5 main categories − artificial surfaces, agricultural areas, forest and semi-natural areas, wetlands and water bodies. Overall, for 944 landscape patches (sum from all years) from Level 1 shape metrics calculations were done. Landscape patches are elementary, further non-divisive units of a landscape and according to [2] are defined as a relatively homogeneous areas that differs from its surroundings. These basic units or areas represent a specific type of land cover and together form a landscape matrix [2]. It is possible to group fundamental landscape patches according to their common characteristics to obtain more general patch type in different scale level, e.g. using CLC nomenclature − artificial surfaces are composed of urban fabric; industrial, commercial and transport units; mine, dumps and construction sites; and artificial, non-agriculture vegetated areas [1]. Furthermore, industrial, commercial and transport units consist of industrial or commercial units, road and rail networks and associated land, port areas and airports, which represent the highest resolution units or patches in CLC nomenclature.

Olomouc region (Fig. 1) was chosen as a study area, with more than 300 patches of CLC nomenclature types in each reference year, in order to follow previous fractal analysis of this area [11]. Olomouc region has an area around 800 km^2 and lies in a valley almost 20 km wide in south-east direction. This part of the region is mostly covered with agricultural areas and artificial surfaces, which are villages and the centre of the region – Olomouc city. North east part of the region is represented by hilly landscape and is covered with forests and semi-natural areas.

Shape metrics are fundamentally based on an area of a shape and its perimeter (these two characteristics are itself considered as shape metrics and are very easy to obtain), but most of metrics are more complicated to calculate and are treated as shape indexes. Anyway, there are plenty of software tools to perform metrics calculation. In this study, FRAGSTATS 4.1 and Shape Metrics toolbox for ArcGIS 10.x for Desktop was used. Multivariate statistics was performed in RStudio environment using R Project programming language.

List of metrics calculated in this study are in Table 1 and their description is available in [8] and in [12]. Nevertheless, it is worth to mention, why it is useful to calculate shape metrics. Since shape metrics take into account only geometric properties of the patch, it is possible to eliminate expert subjectivity in landscape description process. There is no doubt that expert skills are crucial in decision making process, but shape metrics serve them as a "statement of fact" to support their expert knowledge.

Prior to the shape metrics computing, their selection needed to be done, because calculation of some metrics is time-consuming − Shape Metrics toolbox requires vector data and since vertexes are necessary for complicated formulas of some metrics, calculation time for one single patch takes more than 10 minutes − and therefore those were excluded from the analysis.

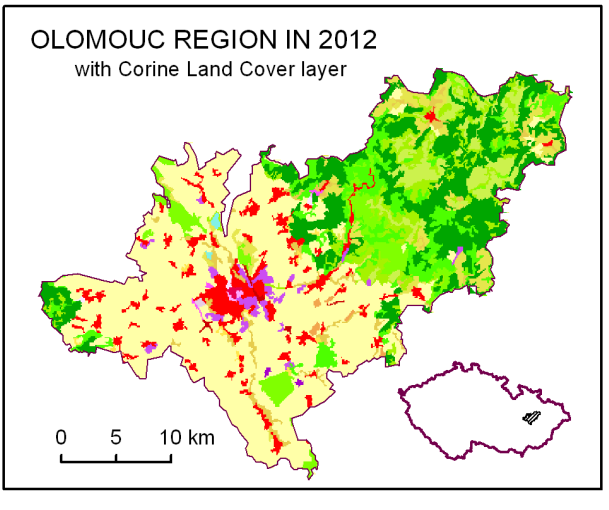


Fig. 1. Current Olomouc region with Corine Land Cover layer from 1990 and its position within Czech Republic

Table 1. Shape metrics used for geocomputation

Shape metrics	Shape metrics
Area index	Girth index
Circumscribing index	Normalized Girth index
Contiguity index	Gyrate index
Core index	Perimeter-area ratio index
Core Area Index	Perimeter index (FRAGSTATS 4.1)
Number of Core Areas	Perimeter index (Shape Metrics Toolbox)
Dispersion index	Normalized Perimeter index (Shape Metrics Toolbox)
Normalized Dispersion index	Proximity index
Depth index	Normalized Proximity index
Normalized Depth index	Range index
Detour index	Normalized Range index
Normalized Detour index	Shape index
Exchange index	Spin index
Normalized Exchanged index	Normalized Spin index

Shape metrics in Table 1 were calculated for every single patch in CLC datasets. Next step was to perform Principal Component Analysis (PCA) of shape metrics to substitute the informational rich complete list and set main three components for consequent clustering. These components are in sum carrying 92 % of the original dataset variability and are composed of various metrics (main variance contribution from Gyrate index, Shape index, Core index, Normalized Core index, Proximity index, Exchange index, Spin index, Girth index, Dispersion index, Range index and Detour index). These and other metrics are forming the first, second and third component with different weights. Principal Component Analysis and the estimation of number of clusters could be depicted via graph of similarity of components within various numbers of clusters (Fig. 2).

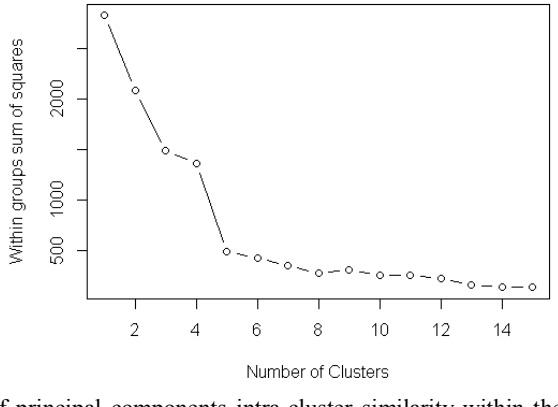


Fig. 2. A plot of principal components intra-cluster similarity within the specific number of clusters

Figure 2 shows a similarity, based on properties of shape metrics, in given number of clusters according to the method of least squares. It is clear that the similarity within 5 clusters is the highest with the respect of cluster number minimalization. The similarity highly increases between 4 and 5 numbers of clusters and does not significantly increase further. Therefore, it is optimal to cluster the dataset into 5 groups which correspond with the CLC Level 1 nomenclature.

Next step was to perform a cluster analysis. To find the best cluster method, a cluster simulation was run. Overall, 840 combinations of methods and individual settings combinations were given. It is quite subjective phase which cluster method and its settings to chose. It depends on what the user desires to achieve. Nevertheless, the simulation of cluster method suitability was performed using silhouette index. The higher the silhouette index the more suitable a clustering method is. There were only marginal differences among silhouette index values of the best proposed methods and that is why the selection of methods was partly left on researcher subjectivity.

Because there are five categories in CLC Level 1 nomenclature and according to withiness of clusters (Fig. 2), only those cluster methods with highest rank in simulation that define five groups were selected.

The first one was hierarchical method (method which creates tree structure − dendrogram) called DIANA − DIvisive ANAlysis Clustering. The DIANA-algorithm constructs a hierarchy of clusters; starting with one large cluster containing all objects and then the cluster is divided until each cluster contains only a single object [6]. Then, the number of groups is defined, and according to that, values are clustered (Fig. 3). For better interpretation and visualization, colour bars were added. Upper bar is representing desired five target clusters, lower bar is depicting five groups of every single patch from CLC Level 1 nomenclature matching to upper bar.

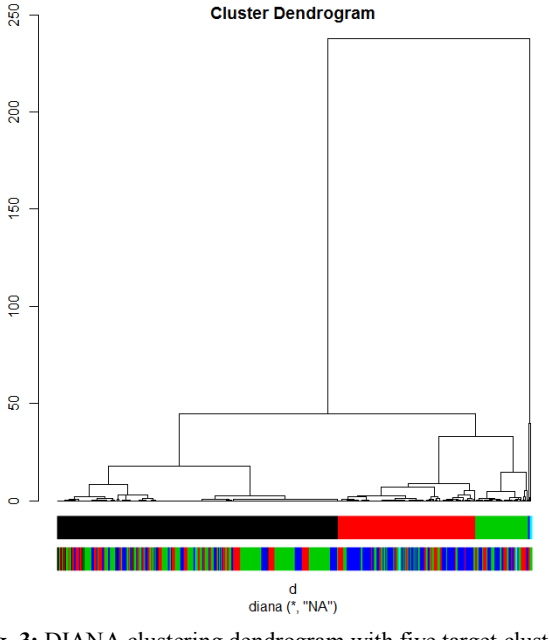


Fig. 3: DIANA clustering dendrogram with five target clusters.

The second method was non-hierarchical, and partitioning, respectively, which means that dataset is broken up into desired number of groups using medoids (representative objects of a dataset, whose average dissimilarity to all surrounding objects is minimized) and is called PAM − Partitioning Around Medoids. This method is similar to the K-means clustering, but K-means uses means or centroids to cluster a dataset. The PAM is treated to be more robust than K-means because of minimizing dissimilarity instead of Euclidean distances ([7], [13]). Resulting clusters according to the two main components are depicted via 2D graph (Fig. 4).

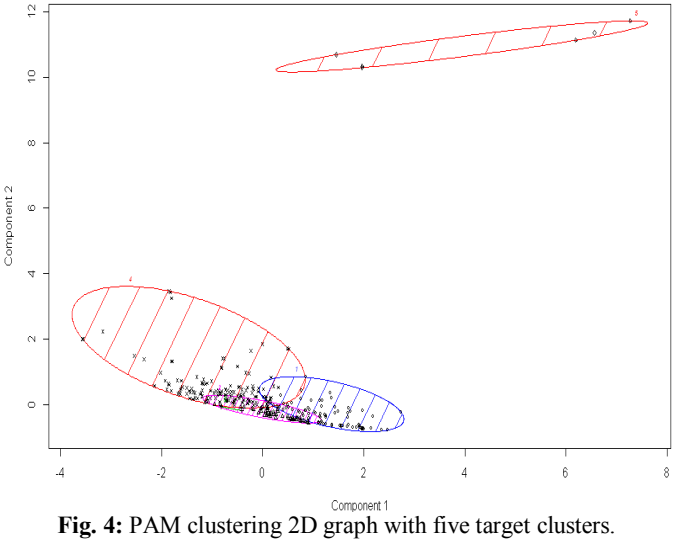


Fig. 4: PAM clustering 2D graph with five target clusters.

3 Results and Comments

Both clustering methods were performed upon shape metrics and their principal components, respectively. Cluster groups were set only according to quantitative values and only non-spatial attribute space of the dataset was performed. Resulting groups are interpreted according to their patch type membership and shape characteristic. Clustering merge patches with the respect of their shape but not directly according to the patch CLC Level 1 type as formerly proposed. Thus, clusters are formed mostly of geometrically similar patches that are, for the most cases, partially patch CLC Level 1 type-independent. Anyway, there are some groups with a significant ratio of one specific patch type category.

First clustering (DIANA) delimitates 5 main clusters (Table 2). Main patch type in the first cluster is agriculture areas (49 %). In the second and third one, main patch type is artificial surfaces (59 %) and (42 %), respectively. Other patch types are not so dominant.

Table 2. Number of patches in DIANA clustering.

Cluster number	Total number of patches	Total number of patch type				
		Agriculture areas	Artificial surfaces	Forest and semi-natural areas	Water bodies	Wetlands
1	560	275	124	136	3	22
2	273	163	67	40	3	0
3	105	27	44	31	3	0
4	3	0	0	3	0	0
5	3	3	0	0	0	0

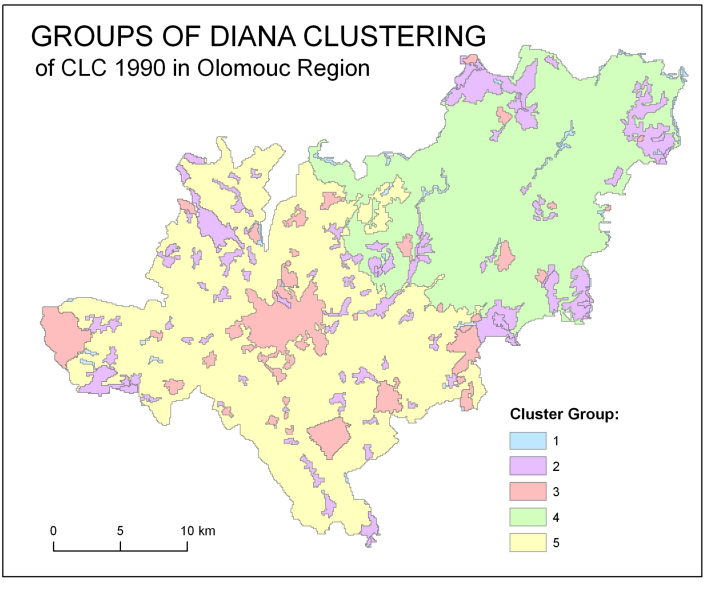


Fig. 5. Olomouc region with Corine Land Cover layer from 1990 and cluster groups according to DIANA clustering method.

Figure 5 shows individual patches classified by DIANA method into 5 groups. For the group number 1 contains mostly very small patches that are close to a minimal size defined by CLC methodology (25 hectars) and are narrowly elongated. Group number 2 incorporates mainly incompact and complex patches (patches with gaps, complicated shapes etc.). Group number 3 is similar to the previous one but patches are more compact (excluding Olomouc city due to its area metrics values) and more regular in their shapes. Groups number 4 and 5 are very similar and are composed of forests and semi-natural areas (Group number 4) and agriculture areas (Group number 5). This is because these two landscape types are represented in GIS as a continuous layer and are extraordinary in all aspects of shape metrics values.

The very same principle as in the previous case was used to CLC dataset using PAM method of clustering. Target clusters defined by PAM are in Table 3. It is evident from both Table 3 and Figure 6 that this non-hierarchichal method distributed patches into groups more equally (excluding cluster number 5).

Table 3. Number of patches in PAM clustering.

Cluster number	Total number of patches	Total number of patch type				
		Agriculture areas	Artificial surfaces	Forest and semi-natural areas	Water bodies	Wetlands
1	191	51	82	46	0	12
2	255	115	62	67	3	8
3	210	127	41	40	0	2
4	282	76	146	54	6	0
5	6	3	0	3	0	0

Main patch type in the first and fourth cluster is artificial surfaces (43 %) and (52 %), respectively. In the second and third one, main patch type is agricultural areas (45 %) and (61 %), respectively. Other patch types are not so dominant.

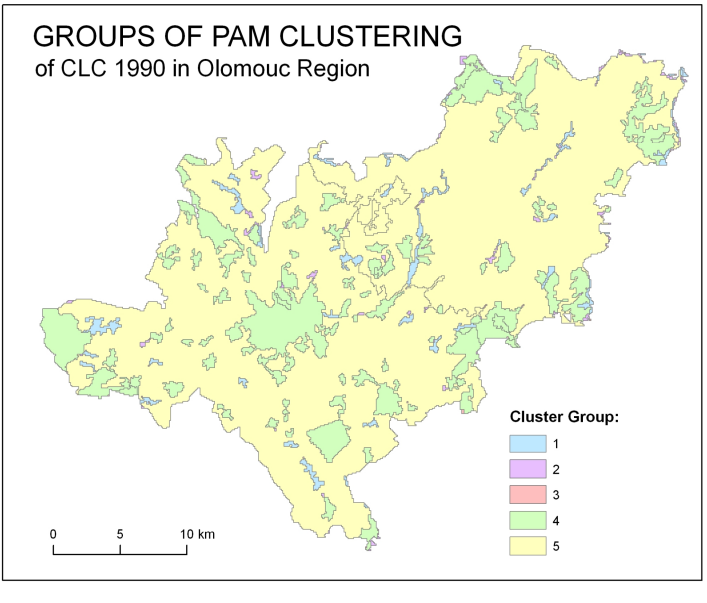


Fig. 6. Olomouc region with Corine Land Cover layer from 1990 and cluster groups according to PAM clustering method

Excepting the group number 1, which is characteristic by containing rather small patches and those narrowly elongated, rest of the groups are the mix of various patches. Forests and semi-natural areas that made up self group using previous DIANA method (group number 4) are now joined with agricultural areas (in DIANA method group number 5) represented in this case by group number 5. Group number 3 contains mainly individual small patches. Barring the group number 1, it is very difficult to find some common characteristics for each group calculated by PAM clustering method. Therefore, it is more suitable in this case to perform analysis of the landscape using DIANA clustering method. However, it depends on the purpose what clustering method to use. If one want to have a complex view onto a landscape, DIANA could be used. On the other hand, PAM identified and pinpointed patches that are narrowly elongated more clearly, thus PAM could serve as a clustering method for elongated patches searching.

Aim of this analysis and calculation was to use clustering methods in order to create distinctive groups of landscape patches. Assumption was that CLC Level 1 patch type is directly influenced by their shape metrics, and vice versa. Ideally, if one of these clustering methods creates same clusters as original types of patches (e.g. artificial surfaces will form their own cluster), it will be very reliable to use them in future automatic classification of any patches. But none of cluster groups in both clustering methods were typical by containing one specific group of patch type in significant amount to claim that e.g. artificial surfaces has very unique shape and thus they form a special group. It is possible to use fuzzy words (e.g. it is more or less "agricultural" cluster) for concluding evaluation statements. Thus, it is needed to analyze patches individually and to search for contexts in detailed level in CLC nomenclature. On the other hand, maybe if larger area would be studied (e.g. entire Czech Republic), the similarity within the cluster would be greater due to the total number of patches involved into shape metrics computation. In other words, proportion of different patch types would not affect final results that much.

Hereby presented procedure could be also modified in the way that input clustering variables will not be principal components, but values of shape metrics themselves. Or another clustering method will be used, regardless to the cluster precision simulation.

Although previously presented results could not provide very convincing results at the first sight, the opposite is true because of the combination of strictly statistical methods together with spatial (visual) evaluation allowed new possibilities of data analysis to arise and unhide clusters of similar areas with similar properties.

Nevertheless, by using above mentioned methods, it is possible to group CLC patches according to their shape similarity, which is useful in a landscape evaluation. Consequent interpretation should take into account the knowledge of shape metrics and the geographic region for which landscape patches are analyzed.

Acknowledgments. The article was created within the project CZ.1.07/2.3.00/20.0170 and CZ.1.07/2.4.00/31.0010, supported by the European Social Fund and the state budget of the Czech Republic.

References

[1] EUROPEAN ENVIRONMENTAL AGENCY (2007): EEA Technical report No. 17/2007, CLC2006 technical guidelines, Copenhagen, 2007, 70 p.

[2] FORMAN, R.T.T (1995): Land Mosaics: The Ecology of Landscapes and Regions. Cambridge University Press, Cambridge, UK

[3] GUSTAFSON, E. J. (1998): Quantifying Landscape Spatial Pattern: What Is the State of the Art?, Ecosystems, Vol. 1, No. 2., pp. 143−156.

[4] HEROLD, M., COUCLELIS, H., CLARKE, K. C. (2003): The role of spatial metrics in the analysis and modeling of urban land use change, Computers, Environment and Urban Systems 29, pp. 369–399.

[5] HUANG, J., LU, X. X., SELLERS, J. M. (2007): A global comparative analysis of urban form: Applying spatial metrics and remote sensing, Landscape and Urban Planning 82, pp. 184–197.

[6] KAUFMAN, L., ROUSSEEUW, P. J. (1990): Finding Groups in Data: An Introduction to Cluster Analysis. Wiley, New York.

[7] KUMAR, P., WASAN, S.K. (2011): Comparative Study of K-Means, Pam and Rough K-Means Algorithms Using Cancer Datasets. ISCCC 2009, Proc .of CSIT vol.1.

[8] MCGARIGAL, K., MARKS, B. J. (2012): FRAGSTATS HELP, University of Massachusetts, 168 p., available from <http://www.umass.edu/landeco>.

[9] MESEV, V. (2007): Integration of GIS and Remote Sensing, Wiley; 1 edition, 312 p.

[10] NUNGESSER, M. K. (2011): Reading the landscape: temporal and spatial changes in a patterned peatland, Springer, Wetlands Ecological Management 19, pp. 475–493.

[11] PÁSZTO, V., MAREK, L., TUČEK, P. (2011): Fractal Dimension Calculation for CORINE LandCover Evaluation in GIS − A Case Study., DATESO 2011, VŠB-TU Ostrava, pp.186-195.

[12] PARENT, J., CIVCO, D., ANGEL, S. (2012): Shape Metrics (presentation). University of Connecticut, ESRI 2009 User Conference, available from < http://clear.uconn.edu/tools/Shape_Metrics/pubs.htm>.

[13] PARK, H. S., JUN, C. H. (2009): A simple and fast algorithm for K-medoids clustering, Expert Systems with Applications, 36, (2), pp. 3336–3341.

[14] SHPUZA, E. (2007): Urban Shapes and Urban Grids: A Comparative Study of Adriatic and Ionian Coastal Cities, Proceedings, 6th International Space Syntax Symposium, İstanbul. 22 p.

[15] TORRENS, P. M., ALBERTI, M. (2000): Measuring Sprawl. CASA Working Paper 27, UCL London, 34 p.

[16] TURNER, M. G. (1989): Landscape ecology: the effect of pattern on process. Ann.Rev.Ec.Syst. 20: pp.171−197.

[17] WU, J. G., JELINSKI, D. E., LUCK, M., TUELLER, P. T. (2000): Multiscale analysis of landscape heterogeneity: Scale variance and pattern metrics. Geographic Information Sciences 6. pp. 6–19.

QuickXDB: A Prototype of a Native XML DBMS*

Petr Lukáš, Radim Bača, and Michal Krátký

Department of Computer Science, VŠB – Technical University of Ostrava
Czech Republic
{petr.lukas, radim.baca, michal.kratky}@vsb.cz

Abstract. XML (extensible mark-up language) is a well established format which is often used for semi-structured data modeling. XPath and XQuery [16] are de facto standards among XML query languages. There is a large number of different approaches addressing an efficient XQuery processing. The aim of this article is to introduce a prototype of an XML database called QuickXDB integrating these state-of-the-art approaches. We primarily describe main concepts of QuickXDB with stress on our XQuery processor. Furthermore, we depict main challenges such as identification of patterns in an input query. Our experiments show strengths and weaknesses of our prototype. We outline our future work which will focus on a cost-based optimization of a query plan.

1 Introduction

XML data model is often considered as a useful alternative to a traditional relational data model. XML data model is more flexible and maintenance of an XML structure is more simple if we work with a very dynamic structure of a database. Therefore, an efficient XML database can be a very useful tool in many real-world projects. Many different approaches have been developed in recent years [1, 5, 6, 12, 18], however, putting all these ideas into a working XML database represents a challenging task with unsolved problems.

Query languages for XML databases such as XPath and XQuery [16] are considered as a de-facto standard. Our work focuses on an efficient processing of XQuery which includes semantics of XPath. There exist several query models which express a core functionality of XQuery. A twig pattern query (TPQ) is the most simple model used by many approaches [1, 5, 9, 17, 19]. A TPQ is represented by a rooted labeled tree (see the TPQ in Figure 1(c)). There is also a more general query model called GTP which includes more semantics of the XQuery language, such as semantics related to the output formatting, boolean expressions or optional parts of a query. These query models represent an important abstraction which enabled designing many efficient algorithms [1, 5, 9, 17, 19] that are independent of a semantic details of an XML query language. However, correct identification of a TPQ or GTP in a XQuery is not straightforward [12].

* Work is partially supported by Grant of SGS No. SP2013/42, VŠB-Technical University of Ostrava, Czech Republic.

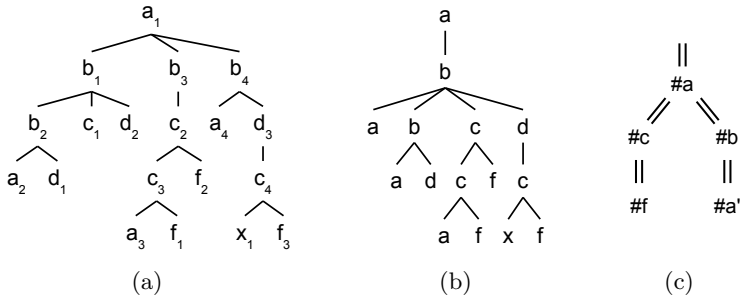


Fig. 1. (a) XML tree (b) DataGuide (c) TPQ

We can find three main areas in the XQuery processing: (1) index data structures and XML document partitioning, (2) join algorithms for a TPQ/GTP processing, and (3) query algebras and cost-based optimizations. Techniques from different areas are often closely related. For example, join algorithms usually need a specific type of index for its efficient run. In this paper, we describe a prototype of an XML database called QuickXDB from the perspective of all these three areas. We describe key concepts of our prototype which enable fast XQuery processing. Main challenges addressed by our prototype are as follows:

- Implementation of an XQuery algebra enabling creation of a query plan, query plan rewritings, and transformation to a physical query plan;
- correct identification of TPQs (more precisely GTP) in a query plan and incorporation of fast and optimal algorithms for the GTP processing;
- cost-based optimizations of a physical query plan that are dependent on XML document statistics as well as on indices available in a database.

The paper is organized as follows. In Section 2, we define basic terms. In Section 3, we summarize basic types of data structures and indices that are available in our database. Section 4 introduces main concepts included in our XQuery processor. In Section 5, we show results of our experiments where we compare efficiency of QuickXDB with other common XML databases. In Section 6, we summarize our results and outline our future work on our prototype of an XML database.

2 Preliminaries

An XML document can be modeled as a rooted, ordered, labeled tree, where every tree node corresponds to an element or an attribute of the document and edges connect elements, or elements and attributes, having the parent-child relationship. We call such a representation of an XML document an *XML tree*. In what follows, we simply write 'node' instead of the correct 'tree node' or 'data node'. An example of the XML tree is shown in Figure 1(a).

The DataGuide tree [8] is a labeled tree where every labeled path from the XML tree occurs exactly once there. Figure 1(b) depicts an example of a DataGuide for the XML document in Figure 1(a).

We assign a label to every node of an XML tree. Node labels allow us to resolve basic XPath relationships between two nodes during the query processing. There are two types of labeling schemes: (1) *element scheme* (e.g., containment scheme [19] or Dietz's scheme [7]) and (2) *path scheme* (e.g., Dewey order [15]). The main feature of labels using a path labeling scheme is that we can extract labels of all ancestors and that they are more flexible during updates.

A *twig pattern query* (TPQ) can be modeled as an ordered rooted tree. Single and double edges between two query nodes represent parent-child (PC) and ancestor-descendant (AD) relationships, respectively (see an example of a TPQ in Figure 1(c)).

3 Indices

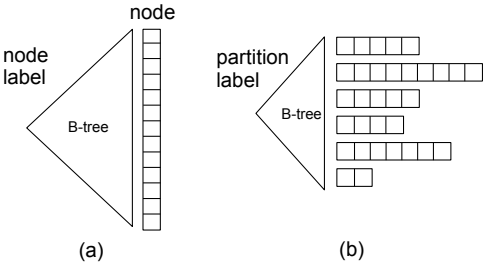


Fig. 2. (a) Document index (b) Partition index

There are two basic type of indices: (1) that having a node label as a key (document index) and (2) that having a node name as a key (partition index). Nodes corresponding to one node name are sorted according to the node label in the case of a partition index. An illustration of these indices can be found in Figure 2.

From the query processing perspective, a document index is very useful if we have a very small intermediate result and we want to use it to process the rest of a query. This type of the query processing can be considered as navigational [11]. However, many join algorithms are based on a partition index. These joins mainly focus on a merging during one sequential read of lists which removes irrelevant nodes. Selection of an appropriate index will be part of future cost-based optimizations [2].

QuickXDB contains also DataGuide, that can be used to speed up the query processing. The main idea is to preproccess the TPQ in a DataGuide and use the result to decrease a search space of a partition or a document index [3].

4 XQuery algebra

In this section, we present a brief introduction to the techniques we use to evaluate real XQuery queries, not only stand-alone TPQs. We use a modified XQuery algebra proposed in [13].

4.1 Processor architecture

There is a block schema of the processor in Figure 3. The traditional first step is to load and parse the input query. The first step yields a syntax tree[1] based on the XQuery grammar standardized by W3C. The next conventional step is usually a normalization phase [12], but we adapted the compilation rules so that the query is directly compiled into the tree of operators (see Figure 5). At the current prototype state, each operator has exactly one evaluating algorithm, so the operator tree has the same meaning as a physical query plan and we call it simply a *query plan*. However, a modular architecture of the prototype does not give us any limitations of using cost-base optimizations to select one of more low-level algorithms evaluating an operator in the future (see Section 6).

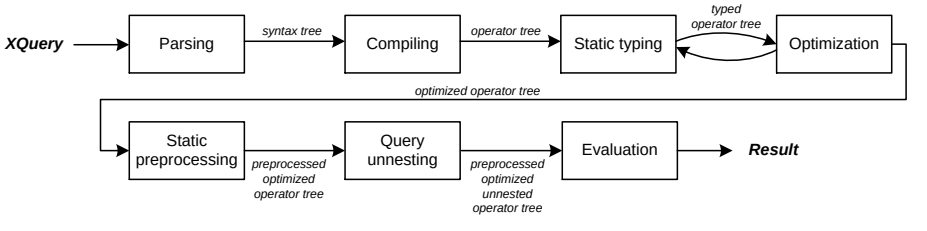


Fig. 3. Block structure of the processor

A compiled query plan is statically typed. The static typing is an important feature for some of the rewriting rules in the next optimization phase (see Section 4.5). None of static pre-computations uses the input XML document. During the optimization a set of rewriting rules is repeatedly applied on the query plan. Each rule searches for a specific pattern with some specific properties and replaces it by a single operator or a subtree of operators. This phase is also responsible for searching the largest possible TPQs in the query plan (see Section 4.6). The crucial attribute of all rewritings is that they are transparent from the result perspective. After applying a rewriting rule, output types of operators have to be reevaluated.

Before the evaluation is done, all the operators are statically preprocessed. For example, we can evaluate a pointer to the algorithm of a function call according to the name of the function. Before the final step we also perform a

[1] Syntax tree is an ordered, rooted, labeled tree, where the nodes stand for terminal / non-terminal symbols of a formal grammar of a language such as XQuery.

query unnesting (see Section 4.4). Finally, the optimized and unnested query plan is evaluated.

4.2 Algebra

A detailed description of an algebra which we use can be found in [13]. Here we only mention some important characteristics to make the following examples clear.

The algebra operates over two different types of sets: (1) a set of *sequences* containing *items* (atomic values or XML nodes), (2) a set of *tables* containing *tuples*. Tuple components are formed of single items or sequences. Furthermore, we have three groups of operators. Operators over sequences, operators over tables, and hybrid operators over both types of sets.

Each operator can have three groups of arguments: independent suboperators, dependent suboperators, and static attributes. Evaluation of the independent suboperators is usually the first step of evaluation algorithms of all operators. The purpose of the dependent suboperators depends on a purpose of particular operator. For example, the *Selection* operator has one independent suboperator computing its input table and one dependent suboperator deciding which tuples of the input table will be passed to the output.

Example 1. Let us consider the example in Figure 5. There are two possible plans of the Q1 query from Figure 4. Both plans lead to the same result. P1a is obtained after the optimization phase without using TPQ rewritings. If we include TPQ rewritings, we obtain the P1b plan. If we run the query against the XML tree from Figure 1, we get the XML node c_1 as the result.

Q1
```
for $x in //b, $y in $x//c
where $x//d and not($y/x)
return $y
```

Q2
```
for $x in //closed_auctions/closed_auction
for $y in //people/person
where $x/buyer/@person = $y/@id
return <out> { $x/type/text() }, { $y/name/text() } </out>
```

Q3
```
for $x in //item
where $x/@id = max(//item/@id)
return $x/name
```

Q4
```
for $x in //europe/item[mailbox/mail/date = "08/05/1999"]
  [description//parlist//parlist][1]
return $x
```

Fig. 4. Example queries

Now let us discuss the P1a plan to outline how the evaluation works. The instances of operators are distinguished by subscripts. Static attributes of operators are given in square brackets. Solid lines represent bindings to the independent suboperators, dashed lines stand for bindings to the dependent ones. The intermediate results of key operators can be seen above them.

The subtree starting with $MapFromItem_1$ represents the first `for` loop of the query Q1. It computes a single-column table with all XML nodes of the name b from the input XML. The $Select_1$ stands for the `$x//d` part of the `where` clause.

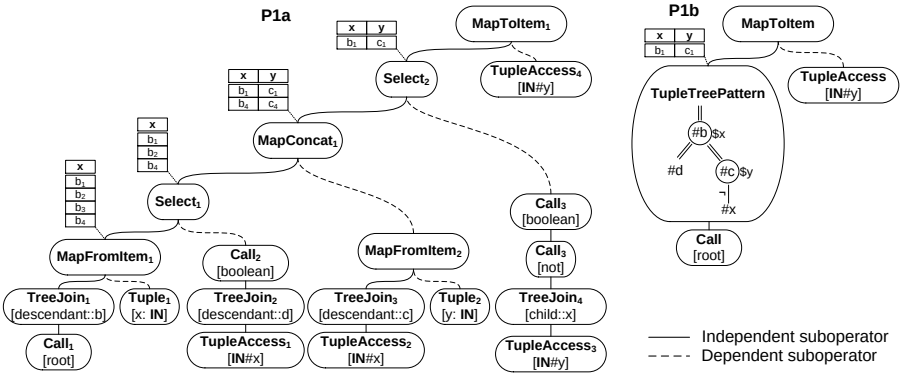


Fig. 5. Example of query plans

Select$_1$ restricts the table from the previous step to only those rows where there exists a d descendant for a particular b. The *MapConcat*$_1$ operator represents the second loop of the `for` clause. *MapConcat* is a crucial operator for this algebra. It evaluates its dependent suboperator for all tuples of the independent input table and joins the evaluated semi-results to the tuples of which they have been computed. Then we can evaluate the `not($y/c)` part of the `where` expression using the *Select*$_2$ operator. The final result is evaluated by the *MapToItem*$_1$ operator. It selects all values of the y column of the input table since we refer to the `$y` variable in the `return` clause.

4.3 Relational rewritings

We can see in the P1a plan that there is no operator evaluating the *and* boolean expression of the `where` clause. The *and* expression is divided into two separated selections: *Select*$_1$ and *Select*$_2$. We can push the selection operators in order to evaluate them as early as possible. This is a well-known rewriting rule from the relational databases. Application of such rule has two advantages: (1) evaluation of the selection as early as possible speeds up evaluation of the entire query, (2) this rewriting extends the possibilities of locating TPQ patterns in the query plan.

There are some other important relational rewritings enabling us to use traditional join operators. In the Q2 query, we can see two independent `for` loops and a typical joining `where` clause. If we use some of the advanced joining algorithms such as hash-join or merge-join, we can significantly speed up the query evaluation. Selection of the proper joining algorithm can be ensured by a cost-based optimization (see Section 6).

4.4 Query unnesting

Note that Q3 is a simple query returning a name of an item with the maximum id. The query can be run over the XMark testing documents [14]. The `where`

clause of Q3 will be compiled into a *Selection* operator. Such operator evaluates the restricting condition over all input tuples carrying individual items in our case. There is a relatively complicated expression computing the maximum id of all items in the right hand side of the equality comparison. Query unnesting means that since an expression is not dependent on any context value (e.g., the `$x` variable of the `for` clause), it can be evaluated only once because its resulting value is always the same.

4.5 Static typing

The static typing step evaluates types of output values of operators in a plan wherever it is possible. Static typing is a crucial feature making some important static rewritings possible. For example, Q4 from Figure 4 contains all three possible types of XPath predicates[2]. The first predicate selects items matching a given equality boolean expression. The second one selects such items where a sequence of XML nodes in the predicate is not empty. The third one passes only the first item matching the above two predicates. Since the purpose of a predicate depends on the type of its expression, there has to be an alternative construction in the unoptimized query plan taking the type into account. However, the output type can be statically evaluated in many cases. Therefore, we are able to rewrite the query plan and use directly one branch of an alternative operator.

Moreover, since there is a possibility to query the order of a node (the last predicate), we need to add (or more technically *map*) an extra column containing sequential numbers. Such column represents a special *positional variable*. If we simplify the query plan knowing we surely do not access the positional variable (the first two predicates), we are able to remove the positional mappings. There are several other rewritings using the statically pre-evaluated types. The goal is to simplify the query plan in order to be able to locate patterns that can be rewritten into a form of TPQ.

4.6 TPQ rewritings

There is still a gap between XQuery algebras and algorithms evaluating TPQs. A lot of algebras have been proposed, but the tree patterns are not supported by them. Only several approaches such as [12] are oriented to search tree patterns in XQuery queries.

A static rewriting of a query plan is the only method how we detect TPQs or more precisely GTPs. Unlike [12], we do not perform any rewritings of the query before the compilation phase. There is a set of rewriting rules searching for the largest possible subtree of operators in the query plan and replacing it by a single tree pattern. A special operator *TupleTreePattern* ensures the use of holistic join algorithms. This operator has been already proposed in [12], but our implementation uses a GTP as its static attribute instead of a linear sequence of path steps.

[2] An XPath predicate is a restricting condition written in square brackets.

The rewriting rules are based on two basic principles: (1) replace single *TreeJoin*[3] by *TupleTreeJoin* with simple tree patterns, (2) merge individual *TupleTreeJoin*s together.

Let us consider the TPQ of the *TupleTreePattern* operator in the P1b plan. All output (circled) query nodes have a reference to the corresponding component of output tuples denoted by $\$q$, where q is the name of the component. We can see that the single *TupleTreePattern* operator in the P1b plan is able to provide the same functionality as the whole subtree of the $Select_2$ operator in the P1a plan. Since the *TupleTreePattern* uses the GTPStack [4] holistic algorithm, the evaluation of P1b is much more efficient for large XML documents than the evaluation of P1a.

5 Experiments

In this section, we compare QuickXDB with some other commonly used XML databases. We choose two standard relational databases Oracle 11g[4] and Microsoft SQL Server 2012[5], three native XML databases Monet DB4[6], Saxon 9.4[7] and BaseX4[8]. Both Microsoft SQL Server and Oracle database have the possibility of indexing XML. Oracle database supports one type of XML index, SQL Server supports one type of primary and three types of secondary XML indices. Also BaseX provides optional XML indexing. Our QuickXDB is able to work with or without any XML index and persistent data structures. In what follows, we write *indexed* or *non-indexed* QuickXDB.

For testing we choose XMark (1.1 GB) [14] and TreeBank[9] (86 MB) data collections and 20 XQueries (10 queries per each collection). A complete set of chosen XQueries can be found in [10]. The queries were divided into two groups according to their purpose: (1) structure oriented queries and (2) content oriented queries.

All experiments were performed on a machine with Intel Xeon E5-2690@2.9GHz processor and Microsoft Windows Server 2008 R2 Datacenter (SP1) operating system. The time of a query execution was the main factor we were focused on.

Every query was run 5 times for each database and each indexing variant. A measured time includes both query execution and query preprocessing (parsing, compiling, etc.). The results are given in Table 1. Each value represents an arithmetic mean of 3 measured times (without the worst and the best case) in seconds.

[3] A *TreeJoin* operator performs elementary path steps in an XML document.
[4] `http://www.oracle.com/products/database/`
[5] `http://www.microsoft.com/sqlserver/`
[6] `http://monetdb.project.cwi.nl/monetdb/XQuery/`
[7] `http://saxon.sourceforge.net/`
[8] `http://www.basex.org/`
[9] XML Data Repository, `http://www.cs.washington.edu/research/xmldatasets/www/repository.html`

	Oracle wo/index	Oracle w/index	SQL Server wo/index	SQL Server prim/idx	SQL Server sec/idx	Saxon	Monet DB	BaseX wo/index	BaseX w/index	Quick XDB non-idx	Quick XDB indexed
Structure oriented XMark queries											
XM1	7.509	DNF	DNF	201.57	DNF	0.374	0.171	1.129	1.203	-	0.066
XM2	DNF	DNF	DNF	DNF	DNF	MEM	E	DNF	DNF	-	0.076
XM3	DNF	DNF	DNF	DNF	DNF	MEM	E	DNF	DNF	-	E
XM4	29.272	DNF	DNF	DNF	DNF	0.837	0.52	1.468	1.444	-	0.47
XM5	24.067	DNF	DNF	DNF	DNF	0.853	0.676	1.843	1.945	-	0.477
Content oriented XMark queries											
XM6	DNF	DNF	DNF	DNF	DNF	DNF	0.394	DNF	18.877	-	10.051
XM7	E	DNF	-	-	-	2.694	-	DNF	DNF	-	8.86
XM8	DNF	DNF	DNF	DNF	51.6	52.942	0.226	252.459	3.141	-	6.271
XM9	E	E	DNF	DNF	DNF	9.048	2.079	9.044	9.409	-	3.829
XM10	DNF	DNF	-	-	-	DNF	0.184	DNF	DNF	-	E
Structure oriented TreeBank queries											
TB1	8.345	DNF	DNF	DNF	DNF	0.385	0.317	1.799	1.808	0.93	0.043
TB2	8.233	DNF	DNF	DNF	DNF	0.494	0.313	2.543	2.655	0.504	0.027
TB3	84.59	DNF	247.662	116.554	358.584	0.281	0.335	11.083	11.246	1.13	0.045
TB4	11.561	DNF	DNF	DNF	148.258	0.499	0.447	2.448	2.239	0.695	0.079
TB5	2.199	DNF	7.072	2.101	407.059	0.759	0.234	1.880	1.724	0.66	0.38
TB6	19.895	DNF	E	E	E	0.25	0.33	7.385	7.242	0.493	0.427
TB7	10.947	DNF	E	E	E	1.492	0.383	8.464	8.308	0.846	0.74
TB8	DNF	E	-	-	-	0.374	1.336	3.058	3.228	0.947	DNF
Content oriented TreeBank queries											
TB9	E	DNF	-	-	-	1.347	0.358	4.897	5.104	0.479	1.746
TB10	1.08	DNF	4.233	2.303	2.189	0.26	0.332	1.236	1.264	0.725	0.52

Table 1. Execution times [s]

The DNF symbol means that a query execution exceeds 10 minutes, E means that a query execution finished with an error. The MEM shortcut stands for the cases when we had to manually stop a query execution due to the protection of the server operating system because of using unacceptably high amount of an operating memory (over 50 GB). A hyphen mark means that a query could not be run because of an unsupported construction or a database was not able to load a data collection.

5.1 XMark queries

The 1GB XMark collection is a representative of large XML documents. This kind of documents are a problem for memory-oriented[10] processors such as Saxon and non-indexed QuickXDB. Saxon processor was able to load the XMark collection, but it consumed over 6 GB space of operating memory. Since the non-indexed QuickXDB (without using persistent structures) is limited to use up to 2 GB of operating memory, it was not able to load the entire XMark document.

Let us see the results of the structure oriented queries (XM1 – XM5) in Table 1. Oracle was able to process 3 of the queries without using index, SQL Server processed only 1 of them using primary XML index. The most problematic query seemed to be XM3 which could not be processed by any of the processors due to the unacceptable high query result.

[10] Memory-oriented processors load the entire XML document into the operating memory and do not use any persistent data structures.

We can observe that indexed QuickXDB outperforms all the other databases for every structure oriented query. The key reason is a detection of a GTP in an XQuery and application of the GTPStack holistic algorithm.

The XMark testing documents contain human-readable data, so querying content may be desirable. Since the current prototype of indexed QuickXDB do not use any content-based index, it performs many random disk accesses. That is the main reason, why the indexed QuickXDB is slower than Saxon and MonetDB in the most content oriented queries (XM6 – XM10).

5.2 TreeBank queries

TreeBank is a relatively small XML document (86 MB) with complex recursive and irregular structure. TreeBank can be loaded into the operating memory by any of the memory oriented query processors.

TreeBank does not have human-readable data so the most of the queries (TB1 – TB8) are structure oriented, where holistic algorithm ensures the fast evaluation. Only 2 of the TreeBank queries (TB9 – TB10) are content oriented, where the non-indexed QuickXDB can give a better performance when compared to indexed QuickXDB.

5.3 Comparison results

We can find four graphs showing relative results of the indexed QuickXDB compared with the other processors in Figure 6. Each value is computed as $100 - 100G$, where G is a geometric mean of relevant quotients t_{xdb}/t_{other}, where t_{xdb} is an execution time of the indexed QuickXDB processor and t_{other} is an execution time of a compared database for the same query. For instance, a value 90 means that the indexed QuickXDB runs ten times faster than the compared database.

Generally we can say that our prototype implementation of indexed QuickXDB runs evidently faster on structure oriented queries. As mentioned before, the key reason is the detection of TPQs and evaluating them by an GTPStack holistic algorithm.

6 Conclusion and future work

In this article, we describe a prototype of our XML database called QuickXDB. We outline the core data structures representing our database and we focus on a description of the XQuery algebra that support a query processing. XQuery algebra enables integration of the state-of-the-art techniques such as holistic joins with other common techniques well known from relational databases. We performed an experiment, where we compare QuickXDB with two major database systems and two native XQuery processors. QuickXDB outperforms all databases for structure oriented queries. As expected, QuickXDB was less successful for

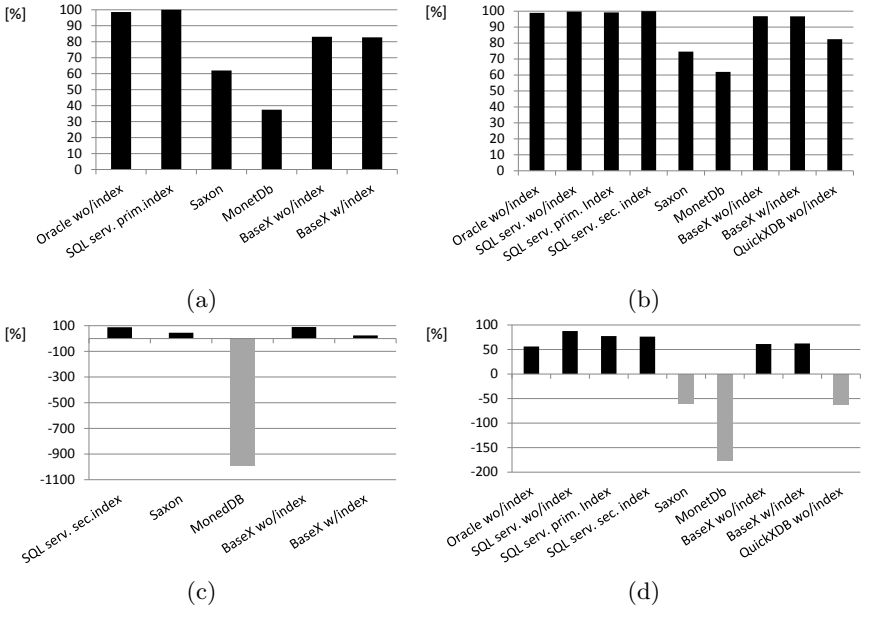


Fig. 6. Comparison of indexed QuickXDB with other XML databases. (a) Structure oriented XMark queries (b) Structure oriented TreeBank queries (c) Content oriented XMark queries (d) Content oriented TreeBank queries

content oriented queries, however, these queries can be accelerated by a common content-based index.

Our algebra contains wide set of rewritings which will support cost-based optimizations in a future extension of the prototype. The major substance missing in our solution are statistics that could help us to automatically select appropriate query plan using all available indices.

References

1. S. Al-Khalifa, H. V. Jagadish, N. Koudas, J. M. Patel, D. Srivastava, and Y. Wu. Structural Joins: A Primitive for Efficient XML Query Pattern Matching. In *Proceedings of ICDE 2002*, pages 141–152. IEEE CS, 2002.
2. R. Bača and M. Krátkỳ. A Cost-based Join Selection for XML Twig Content-based Queries. In *Proceedings of the 2008 EDBT workshop on Database technologies for handling XML information on the web*, pages 13–20. ACM, 2008.
3. R. Bača and M. Krátký. On the Efficiency of a Prefix Path Holistic Algorithm. In *Proceedings of Database and XML Technologies, XSym 2009*, volume LNCS 5679, pages 25–32. Springer–Verlag, 2009.
4. R. Bača, M. Krátký, T. Ling, and J. Lu. Optimal and efficient generalized twig pattern processing: a combination of preorder and postorder filterings. *The VLDB Journal*, pages 1–25, 2012.

5. N. Bruno, D. Srivastava, and N. Koudas. Holistic Twig Joins: Optimal XML Pattern Matching. In *Proceedings of ACM SIGMOD 2002*, pages 310–321. ACM Press, 2002.
6. S. Chen, H.-G. Li, J. Tatemura, W.-P. Hsiung, D. Agrawal, and K. S. Candan. Twig2Stack: Bottom-up Processing of Generalized-tree-pattern Queries Over XML documents. In *Proceedings of VLDB 2006*, pages 283–294, 2006.
7. P. F. Dietz. Maintaining order in a linked list. In *Proceedings of 14th annual ACM symposium on Theory of Computing (STOC 1982)*, pages 122–127, 1982.
8. R. Goldman and J. Widom. DataGuides: Enabling Query Formulation and Optimization in Semistructured Databases. In *Proceedings of the 23rd International Conference on Very Large Data Bases, VLDB 1997*, pages 436–445, 1997.
9. J. Lu, T. Chen, and T. W. Ling. Efficient Processing of XML Twig Patterns with Parent Child Edges: a Look-ahead Approach. In *Proceedings of ACM CIKM 2004*, pages 533–542. ACM Press, 2004.
10. P. Lukáš, R. Bača, and M. Krátký. QuickXDB: A Prototype of a Native XML DBMS. *Technical report No. CS/DBRG/2013-001*, 2013, `http://db.cs.vsb.cz/TechnicalReports/CS-DBRG-2013-001.pdf`.
11. N. May, S. Helmer, and G. Moerkotte. Strategies for query unnesting in XML databases. *ACM Transactions on Database Systems (TODS)*, 31:968 – 1013, September 2006.
12. P. Michiels, G. Mihaila, and J. Siméon. Put a Tree Pattern in Your Algebra. In *Proceedings of the 23th International Conference on Data Engineering, ICDE 2007*, pages 246–255. IEEE, 2007.
13. C. Re, J. Siméon, and M. Fernandez. A complete and efficient algebraic compiler for XQuery. In *Data Engineering, 2006. ICDE'06. Proceedings of the 22nd International Conference on*, pages 14–14. IEEE, 2006.
14. A. R. Schmidt et al. The XML Benchmark. Technical Report INS-R0103, CWI, The Netherlands, April, 2001, `http://monetdb.cwi.nl/xml/`.
15. I. Tatarinov, S. D. Viglas, K. Beyer, J. Shanmugasundaram, E. Shekita, and C. Zhang. Storing and Querying Ordered XML Using a Relational Database System. In *Proceedings of ACM SIGMOD 2002*, pages 204–215, 2002.
16. W3 Consortium. XQuery 1.0: An XML Query Language, W3C Working Draft, 12 November 2003, `http://www.w3.org/TR/xquery/`.
17. H. Wang, S. Park, W. Fan, and P. S. Yu. ViST: a Dynamic Index Method for Querying XML data by Tree Structures. In *Proceedings of the ACM SIGMOD 2003*, pages 110–121. ACM Press, 2003.
18. A. M. Weiner and T. Härder. Using Structural Joins and Holistic Twig Joins for Native XML Query Optimization. In *Advances in Databases and Information Systems*, volume 5739 of *LNCS*, pages 149–163. Springer - Berlin Heidelberg, 2009.
19. C. Zhang, J. Naughton, D. DeWitt, Q. Luo, and G. Lohman. On Supporting Containment Queries in Relational Database Management Systems. In *Proceedings of ACM SIGMOD 2001*, pages 425–436, 2001.

Efficient in-memory data structures for n-grams indexing

Daniel Robenek, Jan Platoš, and Václav Snášel

Department of Computer Science, FEI, VSB – Technical University of Ostrava
17. listopadu 15, 708 33, Ostrava-Poruba, Czech Republic
{daniel.robenek.st, jan.platos, vaclav.snasel}@vsb.cz

Abstract. Indexing n-gram phrases from text has many practical applications. Plagiarism detection, comparison of DNA of sequence or spam detection. In this paper we describe several data structures like hash table or B+ tree that could store n-grams for searching. We perform tests that shows their advantages and disadvantages. One of neglected data structure for this purpose, ternary search tree, is deeply described and two performance improvements are proposed.

Keywords: n-gram, ternary tree, B+ tree, hash table

1 Introduction

N-gram is a sequence of elements, i.e. words in document or words in phrase. These n-grams are used within text operations or text comparisons. It mainly goes about finding plagiarisms, spam detection or comparison of sequences of DNA.

The first problem that occurs within a text comparison is the extraction of n-grams itself. It is generally solved by floating window from the beginning to the end of the document. By extraction it is needed to eliminate duplicated n-grams, store the frequency of their appearance or their position in the document for further comparison.

After finishing the extraction it is needed to look up in the n-grams database. Searching has to be quick even though the amount of data is within gigabytes. Sophisticated data structures were invented for this purpose to provide effective access to searching.

In the following article, the use of the ternary search tree (TST) is described as a data structure to search n-grams. There are tests made to compare ternary search tree to the other commonly used data structures for indexing n-grams.

2 Related Work

The text n-grams extraction is the first part needed for the future use. We are not interested about all n-grams but the specific ones that occur in text at least m-times [8]. It's because we're comparing similarity of documents, respectively the mostly repeated parts of them. In case of huge texts such as 1.5T TREC ClueWeb-B, the use of the ordinary data structures is, such as hash table or search trees, mainly ineffective because the amount of the data cannot be stored in the RAM. Hard drive can be used as a temporary storage where the preprocessed data can be stored [4]. The second option is to utilize structures like a B+ tree or Hash table to manage this amount of data [6].

Within the extraction is also mainly stored the information about n-gram position in the document. To save space, it is appropriate to store this information without redundancy. The use of double indexing for this case was shown within data collections PROTEIN-10M, PROTEIN-100M and PROTEIN-1G. Due to the size of the index was reduced 1.9 to 2.7 times and the search speed increased up to 13 times [5].

One opportunity how to process the n-grams is to store complete text of this n-gram in a data structure [3]. Effective tool for storing the data is for example the ternary search tree [10] in which every node stores information about one n-gram character. As shown by tests on collections Google WebIT and English Gigaword corpus is the data structure fast enough [3].

However, storing whole n-grams in a data structure considerably increases memory requirements. For this case it is better to use two data structures where the words in n-grams are at first converted to unique numbers and only after that the numbers are processed by data structure [1,6]. The most used data structure to map the words to numbers in n-grams is the hashmap [11]. The hashmap is, thanks to its properties, fast enough and memory effective to convert words to numbers. It is ideal in cases where there is beforehand known the word count.

To store n-grams or the words indexes contained in them is widely used B+ tree [1]. It is no wonder because this data structure was designed to search effectively also with regard to the lack of the memory. In every cell of the B+ tree is stored whole n-gram, which is used for comparison during the search process [2].

This attitude was tested on data collection WebIT 5-gram corpus, which contains over 88GB data separated to collection of unigrams to 5-grams. Thanks to word indexing and the use of B+ trees it was managed to store the whole data collection on 598 MB of memory [1]. In this case there is no problem to have the data in memory and thus avoid using slow hard drives. The creation of the indexes for 5-grams itself takes approximately an hour but it lasts only 2 seconds to look up 1,000 5-grams.

One of the key requirements to look up n-grams is the opportunity to use wildcard placeholders, for example when is suitable to look only for particular similarity. When indexing both words and n-grams is first necessary to find a range of words in the first

index. However, this is only possible when the indexes are sorted with the words. If this case is fulfilled, it is easy to look up using data structures like B+ tree [1].

3 Data Structures

There is a huge amount of data structures, which use the pair <*key, value*> to store data. They are mainly called the map. The selection of ideal data structure is not quite easy task. Mainly there is also need to account the type of data, which will be stored simultaneously with the data structure concept.

The array of pairs <*key, value*> can be presented as the easiest data structure. To find the required element it is needed to go through the whole array of elements or the half in the average case. This access is, however, waste of the computing resources.

For faster search the binary search can be used. To use it, we need that the array of elements is sorted. In case of adding one element in the array there is need to move the half of the array elements in average.

There were more complex data structures invented, which are far more effective when inserting new element in the array or looking up one from the array. These will be described in the following paragraphs.

3.1 Hash Table

Hash table is a data structure, which associates the value of the key with the required value. The straight access in the array is used to get the value. The hash is used as the index, which is computed from the key.

Algorithm of hash computation cannot be easily deduced[1]. It has to have many properties, which ensure that this data structure will be enough effective. The key requirement is, that the probability of the same hash appearance is minimal for the given data. Furthermore, the resulting hash has to be in a range of the array size. It is computed by the modular arithmetic [7].

The data table is composed by array, whose elements are the pairs <*key, value*>. However, there is a pointer to the pair stored more often. It is appropriate because the hash table is not always filled, so in these cases the free cells would only occupy memory.

In case that the given hash exists in the data table but for the different key, there are two methods to solve such a collision. The first of these uses a concatenation of stored pairs to form a linked list. In case of looking up there is every hash of the key tested until the agreement occurs or the end of the table is reached.

[1]For the following tests there is the djb2 algorithm used to hash the text

The second attitude is so called open addressing, which computes an alternative position for the given hash up to time when the position is free. In this case there is need to go through all of the alternative space until the given key is found. For this attitude the table has to be larger than the count of the elements.

Whereas the hash function is used for indexing, the n-grams can be indexed by only ordering unigrams one by one. The hash is then computed out of these concatenated unigrams.

3.2 B+ Tree

B+ tree is a tree structure outgoing of B-tree. The only main difference is that B+ tree has values stored only in leaves. Every node of the tree contains the array of the keys and the array of the pointers to the following node.

Using the sequence searching or binary search, the pair of keys is found, which limits the search key. Its index is used to found the next node. This attitude is used to get to the leave, which contains a reference to the value of the given key.

Requirements to B+ tree can be summarized to the following 4 points:

- Root has *N* children at maximum
- Every node besides root has at the maximum *N* and at the minimum *N/2* children
- Data are stored only in leaves
- All the leaves has equal level, they are in the same depth

By fulfilling these requirements the tree structure is formed. This structure is always balanced. The advantage of tree structures derived of B-trees is, that by storing the data to the hard drive, the size of the node can be adapted to the hard drive sector size. N-grams can be stored as sequences in B+ trees. The disadvantage of this attitude is the need always to compare the same prefixes of sequences during the key comparisons, which decelerates the search itself.

3.3 Ternary AVL Tree

Binary search tree (BST) is a data structure, which consists of vertices, which always contains the value and edges. It exists one root element and every vertex contains two edges to the following vertices. One edge points to the vertex, whose value is always bigger and the second edge points to the vertex whose value is smaller.

To look up an element it is enough to start at the root and with the simple comparison go through the tree to the required result. Thanks to this, the searching in the tree achieves an average complexity $O(\log_2 n)$, where n substitutes the number of the tree elements.

During the tree creation the undesirable situation can happen due to miserably ordered data when the linked list is created instead of tree. For example when the data are ordered ascending by value, the tree is created in which every node has only right child. The created tree has complexity of searching defined as *O(n/2)* in average. Mainly this extreme case does not occur but unbalanced tree has far worse time for look up of elements than a balanced tree. This problem can be solved using one self-balancing tree, for example AVL tree [9]. It goes about binary search tree, which in addition fulfills the condition that the length of the left and the right subtree of node differentiates by 1 at maximum. This is ensured by subtree rotation when inserting new node when needed.

This access increases the time severity when inserting and deleting nodes but it ensures more effective searching when inserting unordered data. Storing n-grams can be done similarly as in case of B+ trees. It means that every node would contain whole n-gram and the text of these n-grams would be compared when searching. But still remains the problem when, at minimum, the identical prefix of the given keys must be compared in given node. Ternary search tree is an adjusted version of binary search tree where every node of the tree contains except two links to the following nodes also one more link. This link points to the root of the next ternary search tree, which contains only a part of the key without the prefix which defines the superior tree.

For example, if we would like to index in ternary search tree the letters “ab”, the first ternary search tree would contain the letter “a” and also contain the link to the second one with the letter ”b”.

As ternary search tree can also be unbalanced implying worse search times, it is suitable to combine this data structure with the self-balancing idea. For example the self-balancing ternary AVL tree can be built, which would have suitable properties for future use.

In some cases there is a need to store created tree on the disc. There is one simple solution. The tree itself is stored in the one dynamic array, so by storing this array and some necessary variables the backup is done. To quickly create original tree is just necessary to allocate new array, copy stored one there and copy stored variables.

3.4 Hybrid AVL Tree

Using the ternary search tree for storing whole n-grams can involve problems with the depth of some binary search trees. We made a test with collection of 3-grams[2], where the counts of the search trees in ternary search tree were detected. Table 1 shows result distribution of binary trees. It was found that more than 3 % of binary trees has depth greater than 4. In addition these binary trees are one of the most used binary trees in the ternary tree.

[2]Random lines extracted from Web 1T 5-gram, 10 European Languages Version 1 collection

One option how to stop creating the binary search trees in case of occurrence so deep tree is to change this tree to trees with multiple roots. This is attained by small hash table, which is placed instead of root of the binary search tree. As a hash function is used only modulo to obtain sufficient search speed. By test was found that adding this hash table is effective at the moment when the depth of the tree is greater than 4.

Table 1. Depth of binary trees in ternary tree

BST depth	1,000,000 n-grams	5,000,000 n-grams	10,000,000 n-grams
1	4,770,413	30,204,256	62,114,350
2	239,038	1,188,767	2,522,381
3	103,559	489,166	1,010,700
4	42,342	196,679	390,481
5	15,693	73,598	139,014
6	4,478	24,277	4,7099
7	983	7,821	14,392
8	82	847	1480

3.5 Double Ternary Search AVL Tree

If we use n-gram as n-tuple of words, the considerable redundancy occurs. Thanks to the redundancy the consumption of the working memory considerably increases and operations made with these n-grams are also slow.

If the n-gram can be divided to more words, the words and the n-grams composed of these words can be indexed independently [1, 6]. Indexing of the words is meant the conversion from the text form of word to the numeric value. Occurrence of the word in the text is repeated and therefore it is suitable have these numbers unique only when the words vary. Thanks to this, the redundancy can be avoided.

If the words are converted to numbers, the n-gram itself does not consist of the text now, but of the indexes of numbers. With the use of this knowledge, the two previously described ternary search trees can be joined. During inserting the n-grams to the tree it has to be divided by a set of symbols. Resulting words are inserted into the first ternary search tree, which stores the unique values of the word.

Every vertex in this tree stores one character as a key. After getting a complete list of indexes of words, the second tree is filled. In the second tree, every vertex stores the index of the given word as a key.

The search process is similar. If no word is found in the first tree, the given n-gram surely not exists. If every word exists, the search process continues to the second tree. Similarly as by ternary search tree, also by double ternary search tree the self-balancing AVL and hybrid AVL trees can be used.

4 The Average Time and Space Complexity of Data Structures

Before the testing itself it is suitable to describe the time complexity and space complexity of described data structures. In the following article the *M* would represent the n-gram count and *N* would represent the number of the words in n-gram and *P* would represent the average length of the word in an n-gram.

Hash table using a good hash function and enough big array has the time complexity for the insert operation of $O(N*P)$. This is true when a hash function goes through the whole sequence during the hash computing. If the element count is unknown, there can occur the situation, when the allocated array of hash table is insufficient and it decreases the efficiency of the data structure. At the moment there is need to reallocate an array of hash table and recalculate the hash for all the elements.

A hash table includes keys and a table with the pointers to these keys. This table is usually greater than the elements count, so the size will be twice as large as the elements count. In the case of sequence storing, there have to be next to the key, the pointer to the possible value. The conclusion is that the space complexity is defined as $O(2*N + M*N*P + M)$.

Searching in B+ tree can be divided in two parts. In the first part there is need to find the right link in the node. Whereas the values are ordered, the binary search can be used to search the value. In the second part we move to the next level of the tree. By searching the tree the time complexity is defined as $O(\log_B(M) * \log_2(B))$ where *B* defines the number of keys in the node.

In case of ternary search tree where every node contains one character, link to the left and to the right subtree and the link to the subtree which represents the next character of the sequence. The time and also the space complexity can be hardly exactly determined because it mainly depends on the count of the identical prefixes.

5 Data Structures Comparison

In this section, tests of previously mentioned data structures will be performed. All data structures will be tested on n-gram collection, which contains 1,000,000 of n-grams[3]. Moreover there are four collections, 2-grams, 3-grams, 4-grams and 5-grams. This allows us to discover behavior of data structures to different size of n-grams. Average length of the n-gram of each collection is shown in Table 2. Only in Double hybrid AVL tree and Double ternary AVL tree there is used technique of separate word and n-gram indexing, that was previously mentioned.

[3]Data collection can be found at http://www.ngrams.info/free.asp

Table 2. Average length of n-grams

	2-gram	3-gram	4-gram	5-gram
Avg. length [characters]	14.01	16.77	20.40	24.65

There will be compared seven implementations of data structures. Each test was performed several times for better accuracy. Tests were performed on computer with 2.0Ghz Core 2 Duo processor and 4GB RAM. Measurements were performed by per-process timer from the CPU.

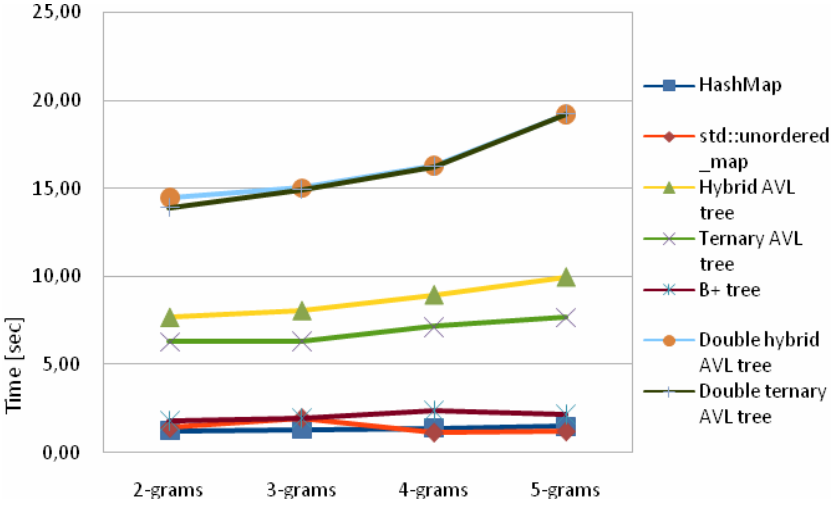


Fig. 1. Insert time comparison

5.1 Comparison by Time of Inserting

This test measures time that is necessary for n-gram insertion. Each data structure is separately created and filled up.

The result on Fig. 1 shows huge difference between ternary tree data structures and the others. This difference may be caused by balancing process due data insertion. This deficiency could be solved by using another type of self-balanced tree, for example red-black tree.

Duration of hash table reallocation seems to be negligible. The worst impact of n-gram size is visible on double trees.

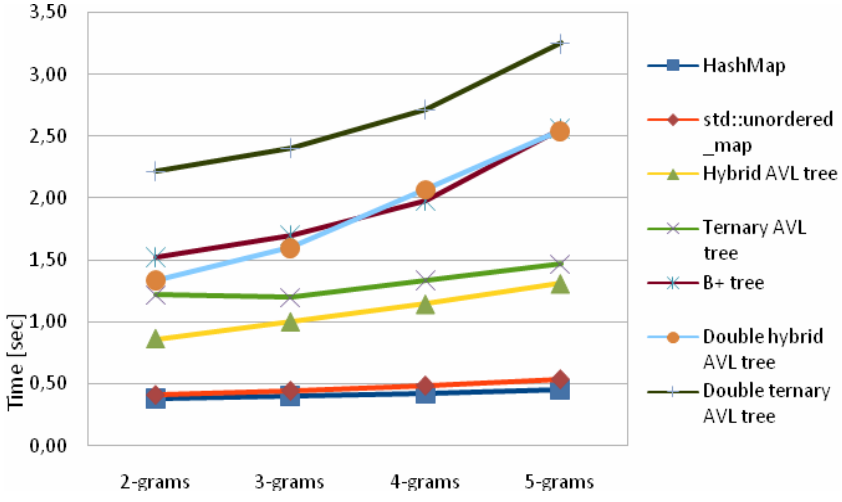


Fig. 2. Comparison of search time

5.2 Comparison by Time of Searching

Search is performed after n-gram insertion. All n-grams from collection are found and time is measured. The result is shown in Fig. 2.

Results shows the best performance of hash table data structures. But on hash table can't be efficiently performed search with wildcard placeholder.

Hybrid variants of ternary trees shows great speed-up. Hybrid AVL tree has up to 29% better search performance than Ternary AVL tree. And Double hybrid AVL tree has up to 40% better search speed than Double ternary AVL tree. Double hybrid AVL tree has comparable results to B+ tree.

The result of B+ tree[4] shows significant increase of search time depending on n-gram size. Moreover, the search time for 5-grams is about 0,41s greater than its insert time. This can be partially caused by necessity of complete look up through the tree in case of search. In the other hand, size of tree increase during insertion.

[4]The implementation of used B+ tree can be found on http://panthema.net/2007/stx-btree/

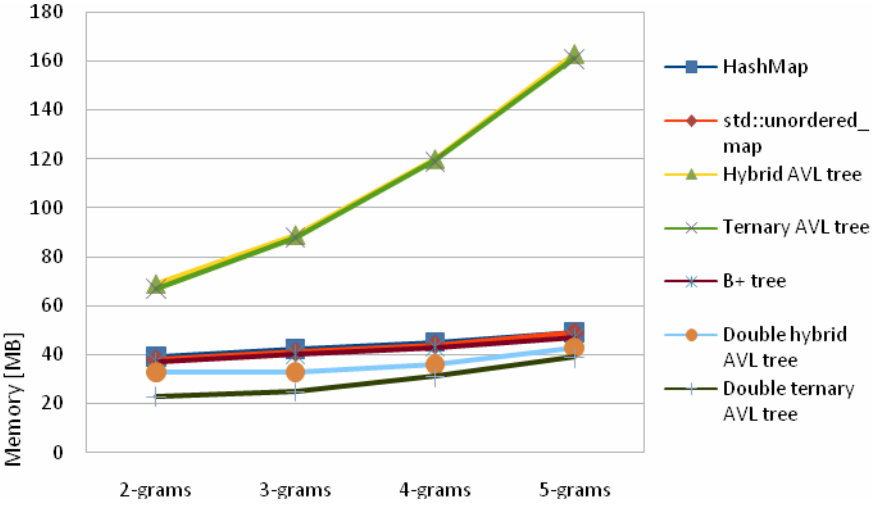


Fig. 3. Comparison of memory consumption

5.3 Comparison by Space Complexity

Last performed test is focused on memory consumption of data structures. Result shows difference of allocated memory before and after data insertion.

Fig. 3 shows large memory requirements of Ternary AVL tree and Hybrid AVL tree, mainly at 5-grams. This is caused by percentage shorter identical prefix of n-grams. This problem solves double variant of this trees.

Double ternary AVL tree has even lesser memory consumption than hash table and B+ tree. The memory consumption is about 40% smaller.

6 Conclusion

This paper described data structures for n-gram indexing such as Hash table, B+tree and ternary trees. Moreover, several approaches for improving ternary search tree efficiency was proposed. The using of the hash table at the 4% nodes of ternary tree with large depth improved the tree efficiency by 40%.

Moreover, this paper shown that the separate indexing of words and n-grams greatly reduced the space complexity. The space complexity of 5-grams reached only 25% originally required memory of ternary tree. The following work will be pointed to the detailed research of data structures for indexing words and n-grams separately. There will also be tested data structures with the requirement to search with the wildcard placeholder. Related to this will be explored data structures for indexing multidimensional data.

Acknowledgement: This work is supported by Grant of SGS No. SP2013/70, VŠB - Technical University of Ostrava, Czech Republic.

7 References

1. Hakan Ceylan and Rada Mihalcea. 2011. An efficient indexer for large N-gram corpora. In Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: Systems Demonstrations (HLT '11). Association for Computational Linguistics, Stroudsburg, PA, USA, 103-108.
2. Douglas Comer. 1979. Ubiquitous B-Tree. ACM Comput. Surv. 11, 2 (June 1979), 121-137.
3. Michael Flor. US Patent, EDUCATIONAL TESTING SERVICE, Princeton, NJ (US). Systems and Methods for Optimizing Very Large N-Gram Collections for Speed and Memory [patent]. United States. Patent Application Publication, US 2011/0320498 A1. Dec. 29, 2011.
4. Samuel Huston, Alistair Moffat, and W. Bruce Croft. 2011. Efficient indexing of repeated n-grams. In Proceedings of the fourth ACM international conference on Web search and data mining(WSDM '11). ACM, New York, NY, USA, 127-136.
5. Min-Soo Kim, Kyu-Young Whang, Jae-Gil Lee, and Min-Jae Lee. 2005. n-gram/2L: a space and time efficient two-level n-gram inverted index structure. In Proceedings of the 31st international conference on Very large data bases (VLDB '05). VLDB Endowment 325-336.
6. Kratky, M.; Baca, R.; Bednar, D.; Walder, J.; Dvorsky, J.; Chovanec, P., "Index-based n-gram extraction from large document collections," Digital Information Management (ICDIM), 2011 Sixth International Conference on , vol., no., pp.73,78, 26-28 Sept. 2011
7. B. J. McKenzie, R. Harries, and T. Bell. 1990. Selecting a hashing algorithm. Softw. Pract. Exper.20, 2 (February 1990), 209-224.
8. J. Pomikálek and P. Rychlý, “Detecting Co-Derivative Documents in Large Text Collections,” in Proceedings of the Sixth International Language Resources and Evaluation (LREC’08). Marrakech, Morocco. European Language Resources Association (ELRA), 2008, pp. 132–135.
9. Robert Sedgewick, Algorithms, Addison-Wesley, 1983, ISBN 0-201-06672-6, page 199, chapter 15: Balanced Trees.
10. David E. Siegel. 1998. All searches are divided into three parts: string searches using ternary trees. In Proceedings of the APL98 conference on Array processing language (APL '98). ACM, New York, NY, USA, 57-68.
11. Justin Zobel, Steffen Heinz, and Hugh E. Williams. 2001. In-memory hash tables for accumulating text vocabularies. Inf. Process. Lett. 80, 6 (December 2001), 271-277.

P system based model of passenger flow in public transportation systems: a case study of Prague Metro*

Zbyněk Janoška and Jiří Dvorský

Department of Geoinformatics, Faculty of Science, Palacký University Olomouc,
17. listopadu 50, 771 46, Olomouc, Czech Republic
zbynek.janoska@centrum.cz, jiri.dvorsky@upol.cz

Abstract. P systems are branch of bio-inspired computing, which takes inspiration from the structure and functioning of a living cell. Current research of P systems focuses mainly on their computational power, applications include biochemical and ecological modeling. In this paper we propose model of passenger flow in metro based on P systems. This model focuses on detailed description of passenger behavior, while retaining simple and robust in description of vehicle flow. Formal description of model is given and simulations using Prague Metro as an example with real traffic flow data from 2008 are presented. Some open problems are discussed and further directions of research are suggested.

Keywords: P systems, passenger flow, Prague Metro, transportation simulation

1 Introduction

P systems were introduced by Păun [8] as s computing model mimicking structure and behavior of a living cell. Păun based theoretical model of P systems on observation, that processes, taking place inside a living cell can be understood as a computation. This theoretical approach to computing is called membrane computing, while formalized mathematical models from this family are called P systems.

P systems consist of three essential parts - membrane structure, multisets of objects and rules, which process them. The structure of P systems is directly inspired by nature, where membranes delimit different compartments of a cell and their purpose is both protection of content of these compartments and also serve as a transportation channel. Different sets of objects are present in these compartments - they are called multisets, because great numbers of each element are assumed inside a living cell. These objects are processed by a set of rules,

* The paper has been completed within the project CZ.1.07/2.2.00/28.0078 InDOG - Innovation of PhD Geoinformatics and Cartography study with support of modern technological trends which is co-financed from European Social Fund and State financial resources of the Czech Republic

which take the form similar to a chemical reaction: $a \rightarrow b$, where a and b are multisets of objects. When a rule is applied, all objects on the left side of a rule are removed and objects on the right side of a rule are introduced into a system. Application of rules is exhaustive, maximally parallel and non-deterministic.
For detailed description of P systems please consult [3], complete biography on P systems is available from [13].
Most research in the field of P systems focuses on computational power (e.g.[9, 11, 12]), applications are restricted mostly to biochemical [1] or ecological [2] topics. In our research, we focus on application of P systems to vehicular traffic flow phenomena [4]. In this paper we propose model for passenger flow simulation in public transportation networks. The model focuses on detailed description of passenger behavior both inside and outside the vehicle; vehicle movement is described briefly and not modeled in detail.
Aim of the model is to accurately estimate numbers of passengers, who use the transportation system at given time. At every time step, numbers of passengers waiting at stations and numbers of passengers in trains are available. This information can be used to examine the performance of the transportation system and the occupancy of trains, for example in case, when a great number of passengers is introduced into a system, or when the schedule of trains is changed. Model can also be used in evacuation studies, when numbers of people in different parts of the transportation system are needed. On the other hand, this model is not suitable for examination of changes of behavior of passengers. It might be of interest to know, how long are passengers willing to wait for a delayed train, or if they rather wait for next train in case, that the current train is almost fully occupied. This kind of research questions requires agent-based modelling, where passengers will make decisions. P systems do not allow decision making – behavior of passengers is described using predefined set of rules, which do not change during the computation, and therefore passengers can not react to ongoing changes within the system.
Performance of a model is shown on an example of Prague Metro, which is simple network of three intersecting lines. Traffic flow data from 2008 are used in simulations.
The paper is structured as follows: in section 2, a formal description of model is given, in section 3, performance of model is examined, section 4 focuses on some problems, which were encountered during the analysis and future directions of research are suggested. Section 5 contains short conclusion.

2 Model description

There exist several levels of traffic modeling, ranging from macro-models, describing traffic flow only in terms of populations of object, to micro-models, where every individual object in the system is examined in detail [5]. For purposes of passenger flow simulation, mezo-scale models are recommended [10]. In mezo-models, some parts of system are described in detail, while description of other parts is rather laconic. Such a model is suitable to focusing on certain part

of traffic flow phenomena, while retaining robust and computationally efficient. In similar manner, proposed model is designed to capture detailed behavior of passengers, while flow of vehicles is brief and simplistic.
Real world system consists of several components, which must be represented in terms of P systems. Metro stations are considered as membranes. Network of stations is represented as a graph, similar to neural P systems [6]. Metro trains are considered membranes, but unlike classical membranes in P systems, they are mobile - their position in the system changes as the system evolves. This evolution is handled by a set of rules [7]. Finally, the passengers are represented as objects. Their behavior follows set of rules, which change according to the position of passengers (inside train or at station).
We believe this representation is very expressive – it is easy to see metro stations as membranes, which are entered and left by passengers – objects. Vehicles serve as membranes too – their function, same as function of living membranes – is to protect its content and serve as a mean of selective transportation (not all objects can enter the membrane and membrane can be entered only on specific occasions). This expressiveness (compared to description using i.e. set of differential equations) together with massive parallelism of model – are two main advantages of P systems for transportation modeling.

Formally, P systems for passenger flow simulation in public transportation networks are following construct:

$$\Pi = (O, l, syn, R), \tag{1}$$

where:

- $O = \{people_{\{a,\ldots,b\}}, empty\} | a, b \in \{l - \{train_1, \ldots, train_n\}\}\}$ is a set of objects, where *people* represents passenger and *empty* represents empty seat inside a train. To each passenger *people*, a sequence $\{a, \ldots, b\}$ is assigned. This sequence is an ordered list of all stations, which passenger visits on his route from station a (current station) to station b (end station of an individual route).
- $l = \{1, \ldots, k, tram_1, \ldots, tram_n\}$ is a set of membranes with k metro stations and n trains.
- $syn \subseteq \{(i, j, t) | i, j \in \{l - \{train_1, \ldots, train_n\}\}, i \neq j, t \in \mathbb{N}\}$ is a set of synapses, representing topology of a network. Each synapse consists of two labels of metro stations i, j and time t necessary to transport train from station i to station j.
- R is a set of rules, which describe the behavior of both membranes and objects inside them. Rules are assigned to membranes, therefore different membranes have different sets of rules and same object in two different membranes can be evolved using different sets of rules. In next section, following notation will be used: train going to station k will be denoted by $[_k\ ]_k$, hence k is label of next, not current station. Each train can be in two states - stopped or moving. Membrane polarization is used to distinguish between

the two, therefore moving train to station k is denoted as $[_k^+ \;]_k^+$. Metro station with label m will be denoted as $(_m \;)_m$. Following set of rules is used to describe the evolution of the system.

1. $people_{\{a,b,\ldots,x\}} \; [_a \; empty \;]_a^- \rightarrow [_a \; people_{\{b,\ldots,x\}} \;]_a^-$ is rule describing passenger entering a train. Passenger, whose next stop is a enters a train going to station a, if there is an empty seat (*empty*) and the train is stopped (negative polarization). Once inside the train, passengers next station changes to b.
2. $[_a \; people_{\{a,b,\ldots,x\}} \;]_a^- \rightarrow [_a \; people_{\{b,\ldots,x\}} \;]_a^-$ is rule describing passenger staying inside a train. Passenger, whose next stop is a and who is already in a train going to a, stays inside and his next stop changes to b.
3. $[_a \; people_{\text{NULL}} \;]_a^- \rightarrow [_a \; empty \;]_a^-$ describes situation, where passenger inside a train has no next station, hence is in his final destination and leaves the system. An *empty* object is created inside a train.
4. $[_a \; people_{\{b,c,\ldots,x\}} \;]_a^- \rightarrow [_a \; empty \;]_a^- \; people_{\{b,c,\ldots,x\}}$ describes passenger leaving the train at transfer station. If passenger, whose next stop is b, is inside train going to a, he leaves the train and *empty* seat appears inside a train. The passenger stays at the current station.
5. $(_i \; [_j \quad]_j^+ \;)_i \xrightarrow{t} (_j \; [_k \quad]_k^- \;)_j$ is rule describing movement of trains inside a network. Moving train in station i, whose next station is j, is moved to station j, its next station is changed to k and the train stops.
6. $(_i \; [_j \quad]_j^- \;)_i \rightarrow (_i \; [_j \quad]_j^+ \;)_i$ changes train from stopped to moving state.
7. $(_i \; [_{\text{NULL}} \quad]_{\text{NULL}}^- \;)_i \rightarrow (_i \quad)_i$ is rule describing situation, where train reaches its final destination (i.e. does not have next stop). Such a train is removed from the system.
8. $(_i \quad)_i \rightarrow (_i \; [_a \quad]_a^- \;)_i$ is rule describing generation of trains in start stations – train going to station a is created in station i. The trains is stopped, therefore passengers can enter the train immediately.
9. $(_i \quad)_i \rightarrow (_i \; people_{\{a,b,\ldots,x\}} \;)_i$ describes arrival of people to the station i. For each passenger, who arrives at the station, a sequence of stations to visit $a, b, \ldots, x$ is generated.

3 Results

Application of model is demonstrated on example of Prague Metro. This transportation system consists of three lines, labeled A (Dejvická – Depo Hostivař), B (Zličín – Černý Most) and C (Letňany – Háje), which intersect at three stations (A–B - Můstek, A–C - Muzeum, B–C - Florenc). Prague Metro consists of 53 stations in total and approximately 1.5 millions of people are transported every day. Trains are in service from approximately 4:40 a.m. to 24 a.m. (midnight) and their frequency changes in time. In five-year periods, survey of passenger occupancy is performed with last survey taking place at 2008. Prague Public Transit Company provided detailed data about transit intensity, which were used in this case study. We selected Prague Metro for its importance as a mean of transportation and also relative simplicity of the system, which allows well-designed simulations.

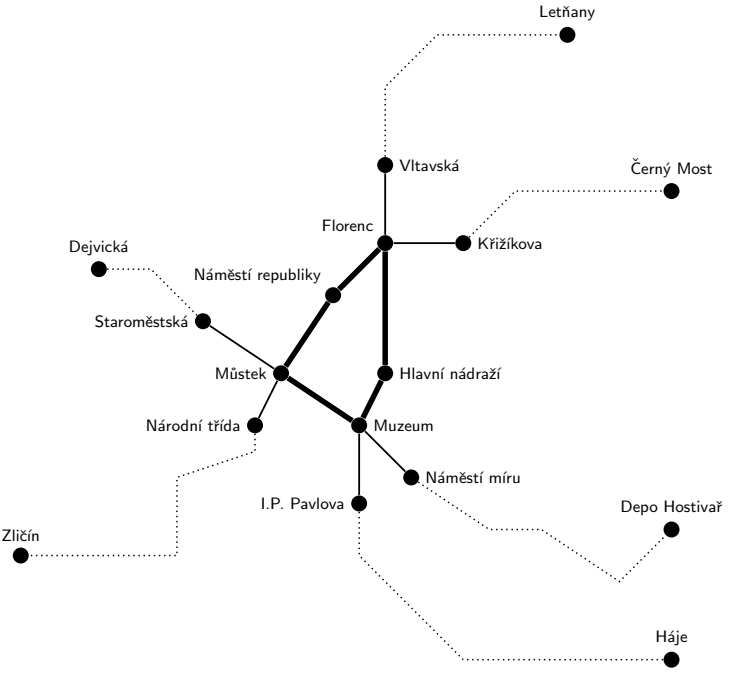


Fig. 1. Prague metro – schematic map

Current (December 2012) schedule of trains was used as basis for generation of trains at start stations. Main problem of public transport modeling is estimation of number of passengers traveling between each pair of stations. Fortunately, this information was provided by Prague Public Transit Company in form of so called Origin-Destination Matrix. Intensity of transport varies during day, which leads to problem with estimation of this intensity. Due to unknown trend of intensities during a day, the quantity was estimated using train schedule. It was assumed, that frequency of train arrivals at given station corresponds with amount of people transported. Kernel density (Epanechnikov kernel, bandwidth=50 minutes) of train frequency was estimated and used as a basis to calculate numbers of passengers entering the system at given time. Figure 2 shows estimated intensity. Values on y axis represent estimated intensity of process, generating "dots" – times of arrival of trains on x axis. We assume, that the process generating times of arrival of trains is in fact the transportation demand of passengers and therefore estimated intensity of this process can be used to calculate the numbers of passengers using the system at given time. Maximal capacity of train was set to 1363, which is occupancy of train 81-71, a standard train in Prague Metro [14]. P systems are discrete in time and one minute was set as a time unit. At every station, the number of passengers N corresponding to estimated intensity function from figure 2 was calculated each minute. Concrete number of passengers was generated from Poisson distribution with $\lambda = N$.

Currently, there are no simulators of P systems, which enable usage of rules in form, which was presented in chapter 2. Set of scripts in python was developed to perform the simulation.

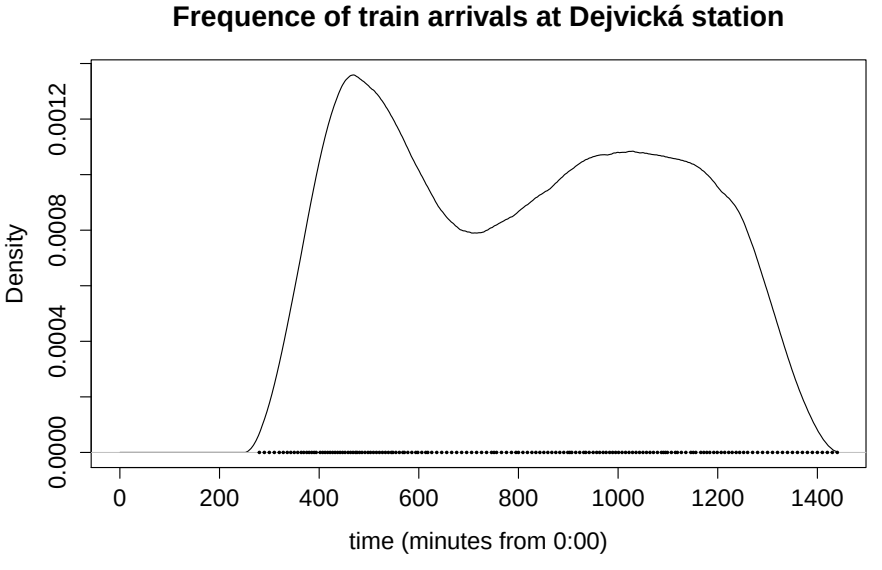


Fig. 2. Estimated train arrival frequency

Four interesting variants of stations were identified in the system and further examined:

1. Final stations – Dejvická, Depo Hostivař, Háje, Letňany, Zličín, Černý Most
2. Transfer stations – Můstek, Muzeum, Florenc
3. Stations between two transfer station – Náměstí Republiky and Hlavní Nádraží
4. Station next to one transfer stations – Vltavská, Křižíkova, Náměstí Míru, I.P.Pavlova, Národní Třída, Staroměstská

3.1 Final stations

Final stations are unique, because only one direction of the train line is served. Simulation showed periodic behavior, where after train departure, no passengers ale left at station (Figure 3). This means, that the transportation demand is fully served.

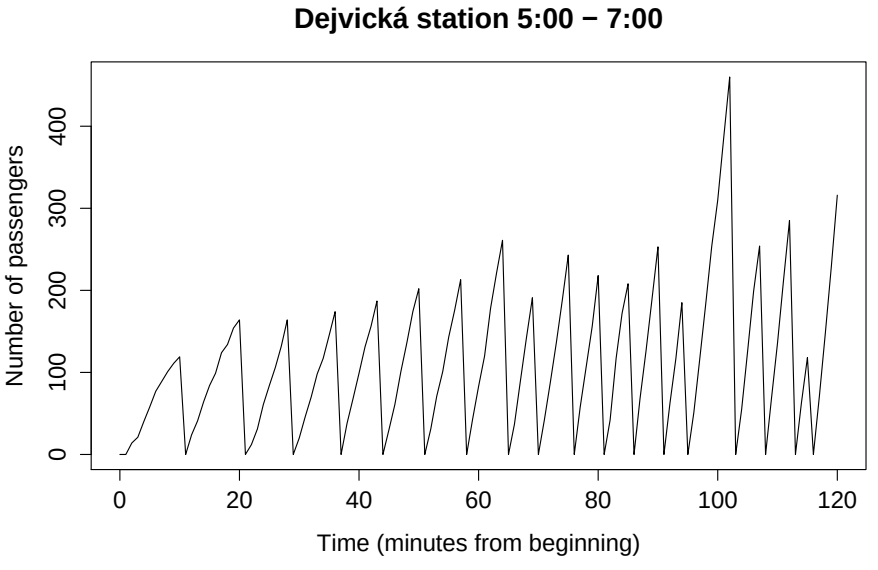


Fig. 3. Number of passengers waiting at the station during 2 hours simulation period

3.2 Transfer stations

There are three transfer stations, where always two lines intersect. These stations are very frequently used and numbers of passengers waiting for train show also cyclic, but more irregular pattern. Moreover, at the beginning of the study period, there is elevated number of passengers waiting for the train (Figure 4). It seems, that the system is not able to handle the demand of passengers for short period of time, but later, the frequency of trains increases and the "wave" of waiting passengers is dissolved. We attribute this behavior not to design of the model, neither we think it represents real behavior of the system, but assume it is caused by incorrectly estimated passenger flow intensity. We will discuss this problem later in chapter 4.

3.3 Stations between two transfer station

Both Náměstí Republiky and Hlavní Nádraží stations show similar, but even more evident pattern as transfer stations. The periodic behavior is more regular and "peak" at the beginning of the study period is more distinctive. Due to high frequency of trains, the numbers of passengers waiting are lower than at most of the other stations (Figure 5).

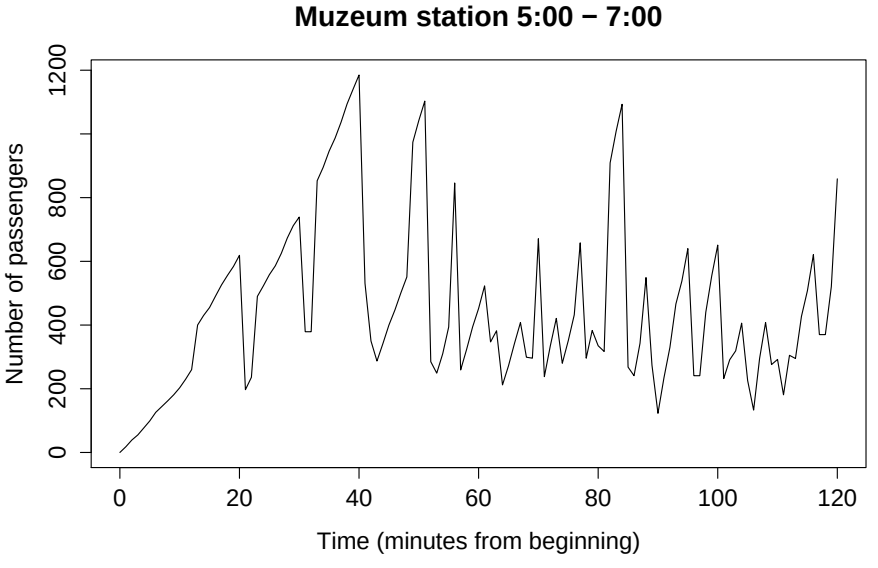


Fig. 4. Number of passengers waiting at the station during 2 hours simulation period

3.4 Stations next to one transfer stations

A rather regular periodic pattern with two peaks can be observed at stations next to transfer stations (Figure 6). Rapid rise in number of waiting passengers was not observed at the beginning of the study period, also the number of passengers returns to values close to zero, which indicates, that the schedule is appropriate to the transport demand. The position of peaks (irregular or regularly spaced) is caused by train schedule and is not caused by stations being immediately after transfer station.

4 Discussion

Possible incorrect performance of the model can result from two types of errors: errors in design of the model and incorrect input values.

Input values of the model are traffic demand and train schedule. Numbers of passengers traveling between all pairs of stations were derived from transportation survey of passenger occupancy and should accurately describe the system. However, only sum of all passengers was available for every station, dynamics of the demand during the day is unknown.
Issue with elevated numbers of passengers at the beginning of the study period

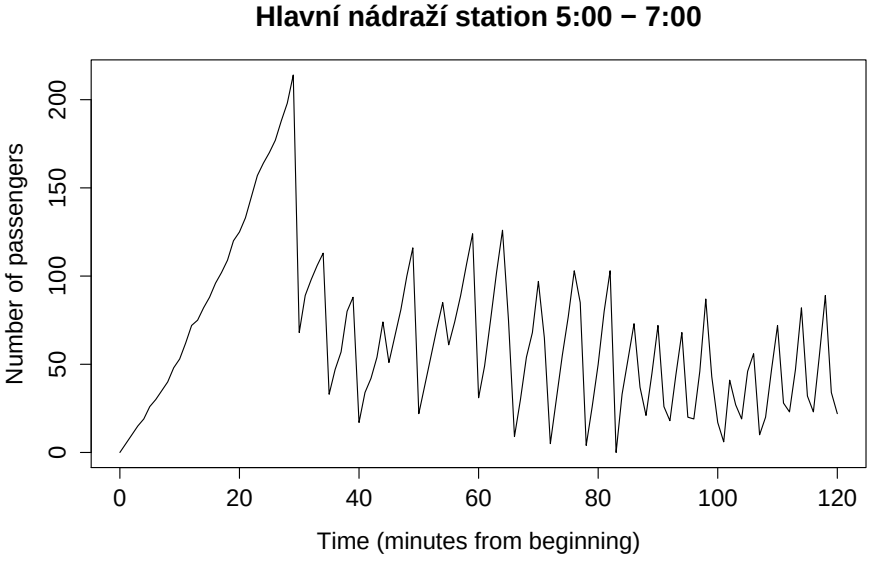


Fig. 5. Number of passengers waiting at the station during 2 hours simulation period

we attribute to incorrect estimation of passenger demand quantity. Different estimates (using different kernels and bandwidths) were examined, however none of them led to results, which both copied global trend and did not produce elevated numbers at the beginning. Correct estimation of traffic demand is crucial step and our next research will be pointed at this direction.
The movement of trains is ruled deterministically using real schedule, obtained from world wide web. In reality, this schedule will be rarely kept, therefore and arbitrary constant can be added to travel time between two consequent stations to account for train delays. This time constant should be preferably generated randomly from known distribution of time delays.

Errors in the design of the model can be represented by ommiting important rules in description of the model. Presented model focuses on more detailed description of passenger behavior, while remaining brief in description of vehicle flow.
Passengers, who exit the train are immediately removed from the system, while in real system, they remain at the station for some time. While not important for examining the capacity or occupancy of traffic system, this might be an issue in i.e. evacuation studies. Extending system by new object - passenger, who no longer participates in transportation, and extending current rules would incorporate this extension, while keeping the model robust and simple.

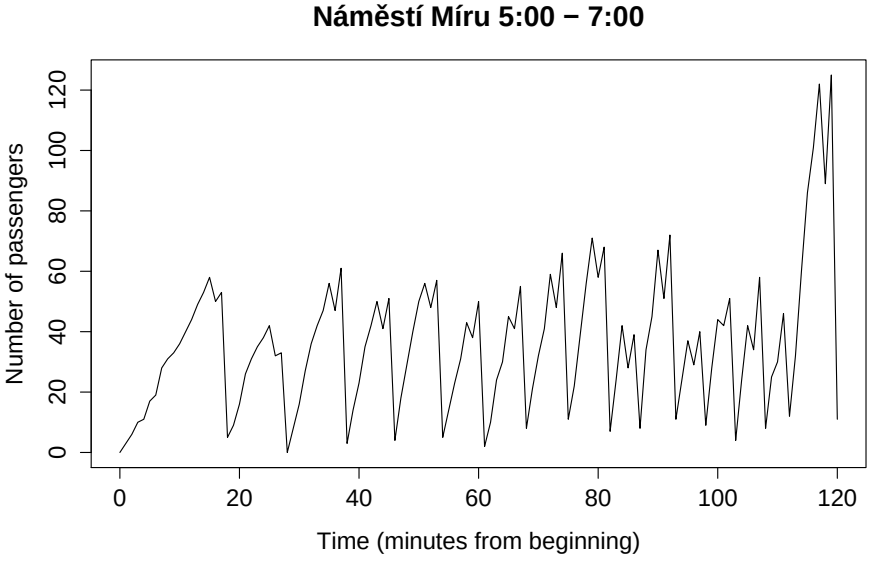


Fig. 6. Number of passengers waiting at the station during 2 hours simulation period

One more possible issue, which is inherent to P systems, should be mentioned. Objects in P systems are not agents, do not posses (artificial) intelligence and do not make decisions. Their behavior is ruled by predefined set of rules, which can be probabilistic and resemble decision making, but essentially, decisions in P systems are not possible and therefore using proposed model for behavioral research of passenger choices would be problematic (i.e. research question "How long are passengers willing to wait for delayed train" is not appropriate for P systems, because requires passengers to make decisions).

The validity of model will be further examined and subjected to complex simulations. Case study only examined numbers of passengers waiting at stations, however occupancy of trains can be of interest for traffic management as well as examination of time, which is spend by certain groups of passengers in the system. Proposed model is discrete both in time and processing units (every passenger is considered an individual element in the system), therefore can be suitable for more sophisticated simulation. Research questions, which will be explored in the future are: How long time delays of trains are still manageable and which length of delays will cause the system to collapse? In the case of change of travel behavior of passengers, which changes could be made to increase the effectivenes of the system? If an increased number of passengers is introduced to the system (i.e. 1000 sport fans going to the game), how will the system respond?

5 Summary

In this paper, a P system based model for passenger flow simulation in public transport systems was proposed. Formal description of model was given and case study using Prague Metro network as example was performed. The case study did not reveal any errors in design of the model, however it became apparent, that correct estimation of numbers of passengers using system at given time is necessary. Cyclic patterns in numbers of passengers waiting at the stations were observed. Open problems associated with usage of P systems for traffic flow simulation were discussed and directions of future research were suggested.

References

1. F. J. Romero-Campero and M. J. Pérez-Jiménez. A model of the quorum sensing system in vibrio fischeri using p systems *Artificial Life 14 (1)*. 95-109 (2008).
2. M. Cardona and M. A. Colomer and A. Margalida and A. Palau and I. Pérez-Hurtado and M. J. Pérez-Jiménez and D. Sanuy. A computational modeling for real ecosystems based on P systems. *Natural Computing 10 (1)*. 39-53 (2011).
3. G. Ciobanu and M. J. Pérez-Jiménez and Gh. Păun. *Applications of Membrane Computing*. Springer, Natural Computing Series, 2006.
4. J. Dvorský and Z. Janoška and L. Vojáček. P Systems for Traffic Flow Simulation. *Lecture Notes in Computer Science 7564*. 405-415 (2012).
5. S. P. Hoogendoorn and P. H. L. Bovy. State-of-the-art of Vehicular Traffic Flow Modelling. *Delft University of Technology*. Delft, 2001.
6. M. Ionescu and Gh. Păun and T. Yokomori. Spiking Neural P Systems. *Fundamenta Informaticea 71 (2,3)*. 279-308 (2006).
7. S.N. Krishna and Gh. Păun. P Systems with Mobile Membranes. *Kluwer Academic Publishers*. Hingham, MA, USA, 2005. 279-308 (2006).
8. Gh. Păun. Computing with Membranes. *Journal of Computer and System Sciences 61 (1)*. 108-143 (2000).
9. A. Păun and Gh. Păun. The power of communication: P systems with symport/antiport. *Journal of Computer and System Sciences 20 (3)*. 295-305 (2002).
10. S. Peeta and A. Ziliaskopoulos. Foundations of Dynamic Traffic Assignment: The Past, the Present and the Future. *Networks and Spatial Economics 1 (1/4)*. 233-266 (2001).
11. P. Sosík. The computational power of cell division in P systems: Beating down parallel computers? *Natural Computing 2 (3)*. 287-198 (2003).
12. C. Zandron and C. Ferretti and G. Mauri. Solving NP Complete Problems Using P Systems with Active Membranes. *Unconventional Models of Computation*. Springer, 2000.
13. P systems web page. `http://ppage.psystems.eu/`.
14. T. Rejdal, www.metroweb.cz. `http://www.metroweb.cz/metro/81-71/81-71.htm`

How can formalization of SOA help in finding solutions for IT systems

Zdeněk Skřivánek, Karel Richta

Dept.of Software Engineering, Faculty of Mathematics and Physics
Charles University, Malostranské nám. 25
118 00 Prague 1, Czech Republic
{zdenek.skrivanek, karel.richta}@mff.cuni.cz

Abstract. Service Oriented Architectures (SOA) are nowadays one of the most important styles in developing new information systems. SOA is attracting a lot of attention in industry as credible tool for managing large infrastructures. These systems divided into divisions have often complex models, which can contain mistakes or are informal. There are not enough current tools for testing semantic correctness of included services. Ways in research of solving such challenges are Model Driven Development (MDD) principles. We introduce the necessity of formalization of SOA in process of developing new systems and also integrating the legacy systems. We want to describe the ideas of how to achieve machine readable specifications using software tools which can then be used to verify the correctness of using the service along the required rules and its testing. We want to open two specific areas of research that is formalization of the transfer process between business and software design models and the formalization of the methods of integrating existing services.

Keywords standardization, interoperability, design, formal methods

1. Introduction

When dealing with large complex systems it is generally recognized that we need appropriate abstractions and structuring principles. Modern enterprises need to respond effectively and quickly to opportunities in today's ever more competitive and global markets.

Service-oriented architectures (SOA) are the latest approach to deliver better understanding and improved techniques to master the complexities of the modern enterprise architectures. SOA is the main architectural concept in the field of service oriented computing. SOA differs from past attempts in several fundamental ways. First, it is language independent and makes no assumption about the underlying programming model. Second, communication is no longer based exclusively on request-respond patterns (RPC/RMI) but the emphasis is on asynchronous events and messages. Third, SOA is complex. SOA sees the development of new applications and ser-

vices mainly as the integration and composition of large scale services and applications rather than as a smaller scale programming problems. These differences arise from the last two decades of solving solutions for IT systems and represent a significant step forward. Definitions of SOA are given by several international bodies/organizations, including the Organization for the Advancement of Structured Information Standards (OASIS) and the World Wide Web Consortium (W3C). The current SOA frameworks offer service reusability, consistency, efficiency and integration. A SOA is a set of components which can be invoked, and whose interface descriptions can be published and discovered. SOA is not only an architecture, rather it is a relationship between the service provider, broker and user. The main advantage of this approach is giving the applications a way to integrate various services available online within the context of the applications specific domain and using them as needed instead of implementing the whole solution from scratch.

The rest of article is organized as follows. In Section 2 we introduce SOA as it progressed through its own history; also we add history of its vital parts. In Section 3, we explain the most important aspect of SOA, and our main interest, the SOA service. In Section 4, we discuss how formalization came into SOA, methods and research steps on this field and related works. In Section 5, we will try to define our point of view on chosen issue, our future work and we will summarize all we wanted to explain in this article.

2. SOA progress through time

SOA emerges from previous successful solutions of developing IT systems and SOA learned & followed the impact of styles such as Modular programming, Model-based development, Software components and Object Orientation methods. SOA also adapts well known technologies as Internet WWW, Open Systems, Net-Centricity, System-of-Systems Engineering and Open Distributed Processing.

From these technologies became integral parts of SOA:

- Web Service Infrastructure
- Message-Oriented Middleware
- Enterprise Service Bus
- Enterprise Application Integration

When taken from the development view SOA adapts:

- Modular programming
- Model-based development
- Software components
- Object Orientation

As times went, business pushed onwards to the software vendors so SOA have to adopt also:

- Loosely Coupled Organization
- Long Tail (Why the Future of Business Is Selling Less of More)

- Mass Customization
- Outsourcing
- Business as a platform
- Enterprise Federation
- Power to the Edge

All these mentioned technologies meets at SOA in its own style as Web Services, Enterprise Mash-Ups, Software as a Service, Real time Enterprise and ESB & Grid.

In our project we will focus on three main aspects of SOA which are important for our future work and they are: XML (content), web services (content transmitters) and their usage in SOA.

Short history of XML: Like HTML, the Extensible Markup Language (XML) was a W3C innovation derived from the popular Standard Generalized Markup Language (SGML) that has existed since the late 60s. This widely used meta-language allowed organizations to add intelligence to raw document data. XML gained popularity during the eBusiness movement of the late 90s. Through the use of XML, developers were able to attach meaning and context to any piece of information transmitted across Internet protocols. Not only was XML used to represent data in a standardized manner, the language itself was used as the basis for a series of additional specifications. The XML Schema Definition Language (XSD) and the XSL Transformation Language (XSLT) were both authored using XML. These specifications, in fact, have become key parts of the core XML technology set. The XML data representation architecture represents the foundation layer of SOA. Within it, XML establishes the format and structure of messages traveling throughout services. XSD schemas preserve the integrity and validity of message data, and XSLT is employed to enable communication between disparate data representations through schema mapping. In other words, you cannot make a move within SOA without involving XML.

Short history of web services: In 2000, the W3C received a submission for the Simple Object Access Protocol (SOAP) specification. This specification was originally designed to unify (and in some cases replace) proprietary RPC communication. The idea was for parameter data transmitted between components to be serialized into XML, transported, and then de-serialized back into its native format. This ultimately led to the idea of creating a pure, Web-based, distributed technology number one that could leverage the concept of a standardized communications framework to bridge the enormous disparity that existed between and within organizations. This concept was called Web services. The most important part of a Web service is its public interface.

Interface is a central piece of information that assigns the service an identity and enables its invocation. Therefore, one of the first initiatives in support of Web services was the Web Service Description Language (WSDL). The W3C received the first submission of the WSDL language in 2001 and has since continued revising this specification. To further the vision of open interoperability, Web services required an Internet-friendly and XML-compliant communications format that could establish a standardized messaging framework. Although alternatives, such as XML-RPC, were considered, SOAP won out as the industry favorite and remains the foremost messaging standard for use with Web services. In support of SOAP's new role, the W3C

responded by releasing newer versions of the specification to allow for both RPC-style and document-style message types.

Completing the first generation of the Web services standards family was the UDDI (Universal Description Discovery and Integration) specification. Originally developed by UDDI.org, it was submitted to OASIS, which continued its development in collaboration with UDDI.org. This specification allows the creation of standardized service description registries both within and outside of organization boundaries. UDDI provides the potential for Web services to be registered in a central location, from where they can be discovered by service requestors. Unlike WSDL and SOAP, UDDI has not yet attained industry-wide acceptance, and remains an optional extension to SOA. Custom Web services were developed to accommodate a variety of specialized business requirements, and a third-party marketplace emerged promoting various utility services for sale or lease [11].

3. SOA Services

Service oriented architecture puts, as the name itself tells, the main pressure on services. Service can implement a single business process or a set of different processes that are made available for integration with other heterogeneous services. Services can be developed using a wide range of technologies, including SOAP, REST, RPC, DCOM, CORBA and Web Services. SOA basically involves three main players: the service provider, the service broker and the service consumer see Fig. 1. The service provider designs and develops a service. The service broker makes this service available to the rest of world through public registries such as Universal Description Discovery and Integration (UDDI) for web services. The service consumer locates the entries in the public registry and binds with the service provider to invoke the web services required.

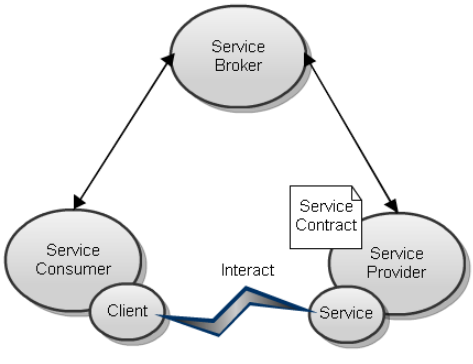


Fig. 1: Overview of requests and response flow between the actors in SOA

Services are described in a standard definition language, have a published interface, and communicate with each other requesting execution of their operations in order to collectively support a common business task or process [13]. Services in SOA are

loosely coupled, supposed to be autonomous, self-contained, one have neither control nor authority over them.

Most common type of service in SOA is a web service which is a method of communication between two electronic devices over the World Wide Web. A Web service is a software function provided at a network address over the web or the cloud, it is a service that is "always on" as in the concept of utility computing. The W3C defines a "Web Service" as "a software system designed to support interoperable machine-to-machine interaction over a network".

It has an interface described in a machine-process format (specifically Web Services Description Language, known by the acronym WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards. The W3C also states, "We can identify two major classes of Web services, REST-compliant Web services, in which the primary purpose of the service is to manipulate XML representations of Web resources using a uniform set of "stateless" operations; and arbitrary Web services, in which the service may expose an arbitrary set of operations."[13].

WS* is set of specifications proposed through W3C, OASIS, WS-I. It uses SOAP, WSDL, WS-Security. It is supported by IBM, Microsoft. It is designed as a technical implementation of service Oriented Architecture.

Service requests are messages formatted according to the Simple Object Access Protocol (SOAP). SOAP entails a light-weight protocol allowing RPC-like calls over Internet [13]. The SOAP request is received by a run-time service (a SOAP “listener”) that accepts the SOAP message, extracts the XML message body, transforms the XML message into a native protocol, and delegates the request to the actual business process within an enterprise. SOAP is by nature a platform-neutral and vendor-neutral standard. These characteristics allow a loosely coupled relationship between requester and provider, which is important especially over the Internet where two parties may resides in different organizations or enterprises.

Requested operations of Web services are implemented using one or more Web service components. Web service components may be hosted within a Web service container providing facilities such as location, routing, service invocation and management. Web containers are similar to J2EE containers. Thread pooling allows multiple instances of a service to be attached to multiple listeners within a single container. Finally the response that the provider sends back to the client takes again the form of a SOAP envelope carrying on XML message [13]. While SOA services are visible to the service client, their Web components are transparent. The service consumer does not have to be concerned with the implementation of the service, as long as it supports the required functionality, while offering the desired quality of service.

Other technology than SOAP nowadays popular is REST. REST defines a set of architectural principles by which you can design Web services that focus on a system's resources, including how resource states are addressed and transferred over HTTP by a wide range of clients written in different languages. If measured by the number of Web services that use it, REST has emerged in the last few years alone as a predominant Web service design model. In fact, REST has had such a large impact on the Web that it has mostly displaced SOAP- and WSDL-based interface design because it's a considerably simpler style to use [13]. A fully REST-compliant architec-

ture is created without using SOAP at all. WSDL version 2.0 offers support for binding to all the HTTP request methods (not only GET and POST as in version 1.1) so it is closer to REST-ful web services [8]. However, support for this specification is still poor in software development kits, which often offer tools only for WSDL 1.1. Complete REST example can be found at [15].

4. Formalization

Formal methods and tools are a popular means of analyzing the correctness properties, specification of a service.

Web services are based on very minimal set of concepts: service, XML document, address and envelope. All the services must expose an interface defined using the WSDL. Several XML-based languages have been proposed for orchestration and choreography. There are many attempts to built formal frameworks for SOA managing orchestration of the services, see [2]. Services orchestration is a key issue in order to fit expectations and reach objects. Amongst the most well known orchestration languages BPEL4WS, XLANG, BPML, WSFL, WS-CDL, pi-calculus etc. The authors distinguish two layers: an abstract layer for which process algebras can be used and a concrete layer using classical services description, orchestration and choreography languages (WSDL, WS-CDL). Services are implemented with programming languages (Java, C# …).

Non-functional properties which include scalability, service reliability, and service flexibility can be assured by Quality of Service (QoS) methods. QoS is the set of techniques to manage network resources. The goal of QoS is to provide guarantees on the ability of a network to deliver predictable results. Elements of network performance within the scope of QoS often include availability (uptime), bandwidth (throughput), latency (delay), and error rate.

Description of service capabilities as automation of composition is addressed by the usage of XML-based standards for a machine readable message and interface description (i.e., WSDL). Also, orchestration languages provide the possibility of defining business processes. Besides these there are some more advised techniques we should also consider as distributed problem solving (DPS).

Each service can be characterized syntactically by its type of input and its type of output messages, i.e., its syntactic interface. The behavior of a service is characterized by the relation of input- and output messages [2]. The service perspective is the most abstract perspective within the SOA framework. The structure within this perspective defines which services are provided at the interface (black-box-view).

There are also commercial successful approaches as The Windows Communication Foundation (or WCF), previously known as "Indigo", is a runtime and a set of APIs (application programming interfaces) in the .NET Framework for building connected, service-oriented applications or IBM WebSphere Service Registry and Repository (WSRR) which is a service registry for use in a Service-oriented architecture etc.

5. Service Model

The service model tells "how the service works"; that is, it describes what happens when the service is carried out [1]. For nontrivial services (those composed of several steps over time), this description may be used by a service-seeking agent in at least four different ways:
(1) to perform a more in-depth analysis of whether the service meets its needs;
(2) to compose service descriptions from multiple services to perform a specific task;
(3) during the course of the service enactment, to coordinate the activities of the different participants; and
(4) to monitor the execution of the service.
The process model identifies three types of processes: atomic, simple, and composite. Each of these is described below.

The *atomic processes* are directly invocable (by passing them the appropriate messages). Atomic processes have no subprocesses, and execute in a single step, from the perspective of the service requester. That is, they take an input message, execute, and then return their output message – and the service requester has no visibility into the service's execution. For each atomic process, there must be provided a grounding that enables a service requester to construct these messages.

Simple processes are not invocable and are not associated with a grounding, but, like atomic processes, they are conceived of as having single-step executions. Simple processes are used as elements of abstraction; a simple process may be used either to provide a view of (a specialized way of using) some atomic process, or a simplified representation of some composite process (for purposes of planning and reasoning). In the former case, the simple process is realized by the atomic process; in the latter case, the simple process expands to the composite process.

Composite processes are decomposable into other (non-composite or composite) processes; their decomposition can be specified by using regular control constructs such as "Sequence" and "If-Then-Else". Such a decomposition normally shows, among other things, how the various inputs of the process are accepted by particular subprocesses, and how its various outputs are returned by particular subprocesses.

6. Related works

A variety of description techniques and formalism already exists, which differ in many aspects such as separation between control, communication, structuring, formal foundation, process composition, concepts and so on. The concept of processes is introduced in Petri nets or in activity diagrams of the Unified Modeling Language. Formalization of the activity diagram semantics is possible in terms of existing formalism such as (Colored) Petri nets or by introducing new formalism. Another Petri net-based approach focusing on the control of flow aspect is YAWL. BPEL is a dominant language for the definition and execution of business process using Web services. Approaches using process algebra like ACP, CCS, CSP and variants in order to formalize work flows.

A wide variety of formal models exists for service-oriented computing. Two distinguished approaches of formalization are presented: process calculus models for expressing and analyzing service based-systems, or models for giving a formal semantic for a standard orchestration language, like BPEL [2]. Business Process Execution Language (BPEL), short for Web Services Business Process Execution Language (WS-BPEL) is an OASIS [2] standard executable language for specifying actions within business processes with web services. Processes in BPEL export and import information by using web service interfaces exclusively. There are three main interactions in web service composition, they are: invoke, send, and receive. In the colored Petri nets they are modeled as transitions [4].

There is need to understand and justify the role of formal engineering methods in developing services for SOA. Address the barriers to deploying formal engineering methods in business. Achieve deployment of formal engineering methods. We want to become inspired from other researchers solutions and we will bring own added value to it.

7. Conclusion and future work

It is clear that there are not enough current known tools for testing semantic correctness of included services. Our aim is to fill this gap with our own solution and programmed tools to use.

Model driven development principles try to achieve machine readable specifications and define software tools which can then be used to verify the correctness of using the services according to the required rules and its testing. The formalization of the transfer process between business models and software design models, and the formalization of the methods of integrating existing services.

There is a gap of abstraction between the formal model and concrete implementation. We should make it as small as possible. First we will seek for a service model. Patterns as components of software development could be used in model driven development or in domain specific modeling (DSM).

Available approaches do not relate available techniques to a basic, comprehensive semantic model. In order to establish an engineering approach for SOA, such a theoretical foundation of the basic concept is needed. Once we establish understanding of concepts, we can start the formalization. Operations can follow as simulation, verification, methodological support, tools etc. and we try to map them to the existing methodologies, tools, and framework if needed.

SOA has various levels of abstraction – similar to object orientation where we distinguish OO-analysis, OO-design and OO-programming. These levels address different aspects like business process modeling, system architecture and implementation. The different levels of abstraction should not exist independently but should be related to each other.

We will try to find out why some projects using formal methods but others not. Identify positive & negative experiences, opportunities & obstacles, what the added value is. And solve all our efforts to the point when integration will be possible to accomplish by non-experts.

Validation and verification must take place in order to ensure the correctness of the solution with the initial business requirements and the defined semantics. Verification and validation is possible only if concepts are clearly defined, their exact relations can be developed. Model of a service is needed. We want to describe SOA scenarios while there can be benefits from the advantages offered by formal methods.

At the business level, we do not want to consider platform-specific aspects but concentrate on core functionalities. Using components and connectors, communicating through dedicated ports only. Define vocabulary of elements as message, channel, semantics, specification, composition etc. Define and constraint relationships, communication mechanisms, and reconfiguration mechanisms. The use of UML and UML profiles as concrete notation for the presented SOA models.

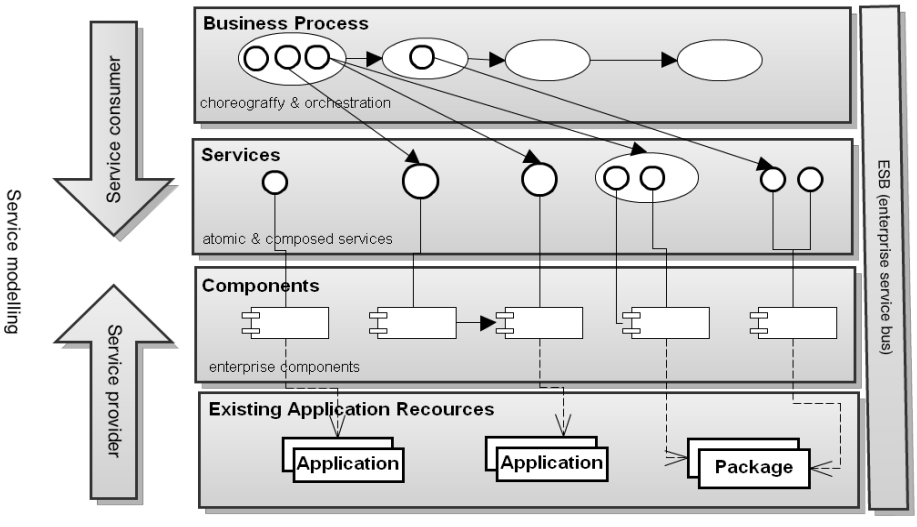


Fig. 2: Overview of service modeling, services and they relationship

Added topic one of the nowadays challenges in SOA could be the gap between transformation of business models with no component specification into a software model. All these steps we will try to finish with dedicated tools to verify of using that service.

This all is meant as verifying the process when creating new services across the needs of business, also one added interest for us will be integration of existing services into SOA by MDI (Model Driven Integration) see Fig. 2. This process of developing services across MDD/MDI we will enchant by adding component such type as a knowledge base based on CBR mechanism (Case Based Reasoning). We argue for the use of the formalization as a basis for the development of tool-supported engineering approach.

References

1. Agarwal, S.: Formal Description of Web Services for Expressive Matchmaking. Ph.D. thesis, University of Karlsruhe, 2007.
2. Allam, D.: A Unified Formal Model for Service Oriented Architecture to Enforce Security Contracts, In: *AOSD*, 2012.
3. Alonso, G.: Challenges and Opportunities for Formal Specifications in Service Oriented Architectures, Springer-Verlag, 2008.
4. Bhakti, M.A.C. – Abdullah, A.B.: Formal Modelling of an Autonomic Service Oriented Architecture. In: *International Conference of Telecommunication Technology and Applications*, 2011.
5. Bocchi, L. – Ciancarini, P.: On the Impact of Formal Methods in the SOA. *ScienceDirect*, 2006.
6. Broy, M. – Leuxner, Ch. – Fernández, D.M. – Heinemann, L. – Spanfelner, B. – Mai, W. – Schlör, R: Towards a Formal Engineering Approach for SOA. *Technical Report*, Technische Universität München, December 2010.
7. Complex Rest example:
   URL: http://www.acme.com/phonebook/UserDetails?firstName=John&lastName=Doe
8. Erl, T.: Service-Oriented Architecture a field guide to Integrating XML and Web services, ISBN 0-13-142898-5, 2009.
9. Erl, T.: SOA Principles of Service Design, ISBN 0-13-234482-3, 2008.
10. Erl, T.: SOA Design Patters, ISBN 0-13-613516-1, 2009.
11. Erl, T.: SOA Kompletní průvodce, ISBN 978-80-251-1886-3, 2009.
12. Khosravi, A. – Modiri, N.: Service Oriented Architecture Essentiality as a Best-Practice for the Development of Large Software Projects. *Journal of Automation and Control Engineering*, 2012.
13. Parazouglu, M.P. - Van den Heuven, W.J.: Service oriented architectures: approaches, technologies and research issues, The VLDB Journal, Volume 16 Issue 3, July 2007 Pages 389 – 415, 2007.
14. Rodriguez, A.: RESTful Web services: The basics, IBM, 2008.
15. Singh, H. – Singh, R.: On Formal Models and Deriving Metrics for Service-Oriented Architecture. *Journal of Software*, Vol. 5,No. 8, 2010.
16. Šelmeci, R. - Rozinajová, V.: One approach to partial formalization of SOA design patterns using production rules. In: *Proceedings of the Federated Conference on Computer Science and Information Systems*, ISBN 978-83-60810-51-4, pp. 1381–1384, 2012.
17. Verjus, H. - Pourraz, F.: A formal framework for building, checking and evolving service oriented architectures, LISTIC – Language and Software Evolution group. In: *ECOWS '07 Proceedings of the Fifth European Conference on Web Services*, pp. 245-254, 2007.
18. Wolff, T.: Using models to design business processes and services, IBM Corporation, 2011.

Comparative Summarization via Latent Dirichlet Allocation

Michal Campr and Karel Jezek

Department of Computer Science and Engineering, FAV, University of West Bohemia,
11 February 2013, 301 00, Plzen, Czech Republic
{mcampr, jezek_ka}@kiv.zcu.cz

Abstract. This paper aims to explore the possibility of using Latent Dirichlet Allocation (LDA) for multi-document comparative summarization which detects the main differences in documents. The first two sections of this paper focus on the definition of comparative summarization and a brief explanation of using the LDA topic model in this context. In the last three sections, our novel method for multi-document comparative summarization using LDA is presented and also its results are compared with the results of a similar method based on Latent Semantic Analysis.

Keywords: comparative summarization, latent dirichlet allocation, latent semantic analysis, topic model

1 Comparative summarization

With the continuing grow of the internet as a source of information, the need for data compression is obvious. This necessity does not apply only to audio or video, but also to textual data (i.e. text summarization). As the amount of textual data grows, the probability of duplicate documents, or documents with very similar features, arises. This is the main problem that we are focusing on in this particular paper and we explore the possibility of utilising the Latent Dirichlet Allocation (LDA) topic model. Comparative summarization is quite a recent area of research and several methods have already been explored. The purpose of these methods is to find some latent information about the input documents and find factual differences between them. These differences are then represented by the most characteristic sentences which form the resulting summaries.

2 Text summarization via LDA

Latent Dirichlet Allocation has already been utilized in several methods, but to our knowledge it has not yet been used in the context of comparative summarization. The closest problem already addressed is the so called update summarization. It aims to search for information, which newly arise in a series of

documents about the same topic. The assumption is that the user is familiar with one document and would like to know what information are additional in another document. We have investigated the already published methods for basic and update summarization using LDA to learn the possibilities of comparing two sets of documents so that we can utilise the best practises to address the problem of comparative summarization.

2.1 Basic summarization via LDA

Latent Dirichlet Allocation (LDA) [4] can be basically viewed as a model which breaks down the collection of documents (the importance of document B for the document set is denoted as $P(D_B)$) into topics by representing the document as a mixture of topics with a probability distribution representing the importance of *j-th* topic for document B (denoted as $P(T_j|D_B)$). The topics are represented as a mixture of words with a probability representing the importance of the *i-th* word for the *j-th* topic (denoted as $P(W_i|T_j)$). This model has already been used for basic summarization in several papers. The topic and word probabilities are in each of the below mentioned methods obtained using the Gibbs sampling method [1]. These summarization methods are briefly described in the following paragraphs. In order to shorten the explanations, only some interesting ideas and explanations (for the purpose of this paper)are mentioned.

The paper [3] has presented new algorithms for scoring sentences based on LDA probability distributions. The basic idea is computing the probability of the r-th sentence from probabilities of words and topics (depending on used algorithm):

$$P(S_r|T_j) = \prod_{W_i \in S_r} P(W_i|T_j) * P(T_j|D_B) * P(D_B) \tag{1}$$

or

$$P(S_r|T_j) = \frac{\sum_{W_i \in S_r} P(W_i|T_j) * P(T_j|D_B) * P(D_B)}{length(S_r)} \tag{2}$$

After obtaining the probabilities $P(S_r|T_j)$, i.e. the probabilities of *r-th* sentence belonging to the *j-th* topic, the selection of the most significant sentences can begin. The process is finished when the number of sentences reaches a predefined amount.

The other paper dealing with LDA-based summarization is [2]. The idea is to combine the LDA topic model and Latent Semantic Analysis (LSA) to reduce the information content in sentences by their representation as orthogonal vectors in a latent semantic space. At first, the LDA probability distributions of topics and words are obtained. After that, for each topic T_j, a term-sentence matrix is created and then the Singular Value Decomposition (SVD) is applied to each of them. The result of the SVD are three new matrices U, Σ and V^T, from which only the third one is utilised. This matrix contains the so called right singular vectors, which basically map topics to sentences. After obtaining the sentence

probabilities, the process of selecting sentences with the best score can run until the predefined summary length is reached.

The paper [8] presents two algorithms for summarization and most importantly a new sentence similarity measure based on LDA. Instead of representing a sentence as a sparse vector using tf-idf, the idea is to use the LDA topic model to represent words and sentences as vectors of topic probabilities. The sentence vector is calculated as an average value of topic vectors of all words in the given sentence. Using this representation, it is a simple matter to measure the similarity between any two vectors using cosine similarity. The summarization algorithms are then based on selecting the best candidate sentence which also has the lowest redundancy with the existing summary until the summary length is reached.

2.2 Update summarization via LDA

The update summarization is the closest problem to ours, so we explored the used methods of comparing LDA topics. The following paragraphs describe methods of update summarization that have been already published and evaluated.

In the paper [6] a novel update summarization framework was proposed. The topics were extracted from two sets of documents A and B by the means of LDA topic model. The topics were assigned into four different categories:

- emerging – topics that newly arise in B
- activating – topics in both set, but with more emphasis in B
- non-activating – topics in both sets, but not too much discussed in B
- perishing – topics only in A

The correlations between old and new topics were then identified with the use of Pearson product-moment correlation. A novel algorithm (CorrRank) was also developed for ranking sentences with topic correlation so that the best ranked sentences can be iteratively added to the resulting summary.

The method proposed in the paper [5] is derived from TopicSum presented in [7] and the topic model of input documents is restricted to only two topics for each document set. The idea is that one topic in each document contains all the already known facts and the second topic contains all the new information that we want to extract.

3 Comparative summarization via LDA

This section will thoroughly describe our novel method for comparative summarization using LDA topic model. Our idea is to use this topic model to represent the documents, compare these topics and select the most significant sentences from the most diverse topics, to form a summary.

The first step is to load the input data from two document sets A and B. The important thing here is that from the perspective of LDA, we treat every

sentence as one document. When we have all the sentences from both sets loaded, we can estimate the LDA parameters (the exact reason will be discussed in the last section of this paper) as follows:

- summaryLength = 10sentences
- numberOfTopics = $\sqrt{\text{numberOfSentences}}$
- numberOfIterations = 3000
- $\alpha = 50/\text{numberOfTopics}$
- $\beta = 200/\text{numberOfWords}$

Before we run the Gibbs sampler (we used the implementation JGibbLDA from [1]) to obtain the LDA topics, we have to remove the stop-words and perform term lemmatization. This way we are sure that there are no words that carry no useful information. With the parameters set and input text prepared, we can obtain the word-topic distributions for each document set and store them in matrices T_A (topic-word) for the document set A and T_B for B, where row vectors represent topics and column vectors represent words. A very important aspect of writing the distributions into matrices is to ensure that both of them have the same dimensions, i.e. to work as well with the words that appear only in one set and including them also in the second matrix (with zero probability). After this, we can compute topic-sentence matrices U_A and U_B with sentence probabilities (we experimented with two equations):

$$P(S_r|T_j) = \frac{\sum_{W_i \in S_r} P(W_i|T_j)}{length(S_r)^l}, \tag{3}$$

or

$$P(S_r|T_j) = \frac{\sum_{W_i \in S_r} P(W_i|T_j) * P(T_j|D_r)}{length(S_r)^l}, \tag{4}$$

where $l \in < 0, 1 >$ is an optional parameter to configure the handicap of long sentences. The row vectors represent topics and the columns are sentences. Next step covers the creation of two diagonal matrices SIM_A and SIM_B which contain the information about similarities of topics from both sets. This is accomplished in two steps:

1. $T_A = [T_{A1}, T_{A2}, ..., T_{An}]^T, T_B = [T_{B1}, T_{B2}, ..., T_{Bn}]^T$, where T_{Ai} and T_{Bi} are row vectors representing topics and n is the number of topics. For each T_{Ai} find red_i (redundancy of i-th topic) by computing the largest cosine similarity between T_{Ai} and T_{Bj}, where $j \in < 1..n >$ and storing value $1 - red_i$ representing the novelty of i-th topic into matrix SIM_A.
2. For each T_{Bi} find red_i (redundancy of i-th topic) by computing the largest cosine similarity between T_{Bi} and T_{Aj}, where $j \in < 1..n >$ and storing value $1 - red_i$ representing the novelty of i-th topic to matrix SIM_B.

Finally, we create matrices $F_A = SIM_A * U_A$ and $F_B = SIM_B * U_B$ combining the probabilities of sentences with the novelty of topics. From these matrices, it is a simple matter to find sentences with the best score and including them

in the summary. For better results, it is essential to compare the candidate sentence with already selected sentences to avoid information redundancy (the comparison is also achieved via the cosine similarity). If a sentence is selected, the relevant vector in F_A or F_B is set to *0* in order to remove the information from the matrix. The final result consists of two independent summaries of predefined length, each of which depicts the most significant information, which are specific for one of the compared document set exclusively.

4 Evaluation

Due to the lack of unified testing data for the task of comparative summarization, we had to create our own data set for evaluation. We have utilised data from TAC 2011 conference to find out if the proposed method brings the expected results. The available data consist of 100 news articles in total, divided into 10 topics, 10 articles each. With these articles, we have created pairs of sets of documents by combining different topics (Figure 1). In every pair, there is one identical topic present in both sets and one topic for each of the sets that are different. This has a simple purpose: to simulate two sets of documents which have something in common, but also some differences. This setup allows us also to easily compute the precision of selecting sentences because we know which sentences we want the algorithm to select. The reason for the use of TAC 2011 dataset is also the fact, that there are three human-created summaries for each of the 10 topics. This allows us to further evaluate our method with the ROUGE toolkit. However, the ROUGE based evaluation is not included in this paper, because it is not yet complete.

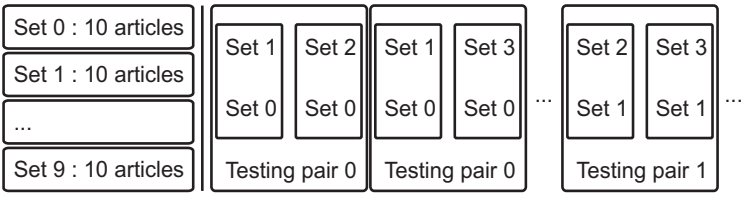


Fig. 1. Creating testing data-sets

Another problem we encountered was how to compare two vectors to gain the best results. We tried two possibilities: cosine similarity and Pearson correlation (as was mentioned in [6]). From these two options, cosine similarity gave better results and comes out as a better choice, even if the precision was only higher in the order of tenths percent.

The last issue of the proposed method is how to set the parameters for the Gibbs sampler to get the best LDA distributions. We have tested our method

on 11 values for both parameters α and β, including values recommended in Section 4 (those depending on the number of sentences or words). Parameter values varied from 0 to 100, and we computed the average precision. The result is on the Figure 2. As can be seen, the α parameter has only a little impact on the precision if the equation 3 is used. On the other hand, for the equation 4, the impact on precision is practically the same as for the β parameter. At the end, the best overall average precision value we were able to achieve was $57,74\%$.

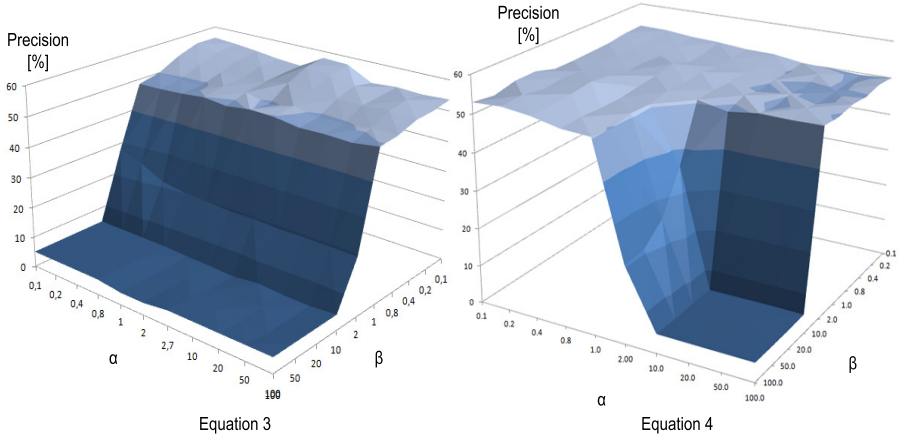


Fig. 2. Average precision depending on parameters α and β for equations 3 and 4

5 Conclusion

In our previous work, we developed a similar method for comparative summarization using Latent Semantic Analysis. In this case, the average precision values were in the range from $61,23\%$ to $98,44\%$ for different configurations of the algorithm. Although the LDA provides more intuitive topic model, it has evidently much lower precision values for any case of given parameters and thus the LSA comes out as a better choice for comparative summarization. The last step in evaluating these two methods is via the ROUGE toolkit, which we are working on right now.

Our future work resides still in the area of comparative summarization, but we would like to explore the possibilities of including sentiment analysis in the process of topic comparison in order to widen the area of usability.

Acknowledgements

The access to computing and storage facilities owned by parties and projects contributing to the National Grid Infrastructure MetaCentrum, provided under the programme "Projects of Large Infrastructure for Research, Development, and Innovations" (LM2010005) is highly appreciated.

References

[1] Xuan-Hieu Phan, Cam-Tu Nguyen. `http://jgibblda.sourceforge.net/`.

[2] Arora, Rachit and Ravindran, Balaraman. Latent dirichlet allocation and singular value decomposition based multi-document summarization. *Proceedings of the 2008 Eighth IEEE International Conference on Data Mining. ICDM'08. Eighth*, pages 713–718, 978-0-7695-3502-9.

[3] Arora, Rachit and Ravindran, Balaraman. Latent dirichlet allocation based multi-document summarization. *Proceedings of the second workshop on Analytics for noisy unstructured text data*, pages 91–97, Singapore, 978-1-60558-196-5.

[4] DM Blei, AY Ng, and MI Jordan. Latent dirichlet allocation. *the Journal of machine Learning research*, pages 993–1022, 2003.

[5] Delort, Jean-Yves and Alfonseca, Enrique. DualSum: a Topic-Model based approach for update summarization. *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 214–223, 2012.

[6] Lei Huang and Yanxiang He. CorrRank: update summarization based on topic correlation analysis. *In proceedings of 6th International Conference on Intelligent Computing*, pages 641–648, 2010.

[7] Haghighi, Aria and Vanderwende, Lucy. Exploring content models for multi-document summarization. *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 362–370, 978-1-932432-41-1

[8] Tiedan Zhu and Kan Li. The Similarity Measure Based on LDA for Automatic Summarization. *Procedia Engineering*, pages 2944–2949, January 2012.

Using Retinex and SVD Algorithms for Detection of Frayed Edge in Steel Plate

Michal Holiš[1], Martin Plaček[1],
Jiří Dvorský[2], Jan Martinovič[2], and Pavel Moravec[2]

[1] Department of computer science, VŠB – Technical university of Ostrava,
17.listopadu 15, 708 33 Ostrava – Poruba, Czech republic,
{michal.holis, martin.placek}@vsb.cz
[2] IT4Innovations, VŠB - Technical University of Ostrava,
17. listopadu 15/2172, 708 33 Ostrava, Czech Republic
{jiri.dvorsky, jan.martinovic, pavel.moravec}@vsb.cz

Abstract. This paper describes a method that tries to improve the accuracy of a machine vision algorithm for frayed edge detection in cold-rolled electrical grain oriented steel plate with usage of the Singular Value Decomposition. The algorithm being improved is based on preprocessing the image with the Multi Scale Retinex algorithm, application of the Sobel filter and additional evaluation logic.

1 Introduction

In our previous paper [6] we have presented an image analysis method for detection of frayed edge in cold-rolled electrical grain oriented steel plate. In this paper we enhance this method with usage of the Singular Value Decomposition (SVD) and present results that were obtained with it. The main idea is to use the SVD to preprocess an image that is being analyzed and try to use it to enhance the image so that it improves detection accuracy.

This paper is divided into three main sections. In Sect. 3 short description of the Retinex algorithm and frayed edge detection is described. Section 4 describes the Singular Value Decomposition. In Sect. 5 frayed edge and detection algorithm are described. Section 6 contains information on experiment design and Sect. 7 contains measured results of the experiment. Last Sect. 8 concludes the paper and summarizes achieved results.

2 State of the Art

Detection of the frayed edge itself is subject that has so far not been studied thoroughly. Articles on the origin of this defect and processes involved in creation of it exist but are focused plainly on the metallurgical side of the subject. As State of the Art in this article a selection of image normalization methods is presented.

© V. Snášel, K. Richta, J. Pokorný (Eds.): Dateso 2013, pp. 87–97, ISBN 978-80-248-2968-5.
VŠB – Technical University of Ostrava, FEECS, Department of Computer Science, 2013.

To enhance the quality of an input image and highlight the frayed edges in the input image many preprocessing methods can be used. In the process of design of our detection algorithm we have tried number of preprocessing method, some of them are described in this section.

2.1 Histogram equalization

The histogram equalization aims to increase global contrast of a processed image to adjust local intensities. This method is useful when an image is composed mainly of close values as it spreads the most frequent values of an image and allows areas with close values to gain much higher contrast. More detailed description of this method can be found in [1].

2.2 Self-Quotient Image

The Self Quotient Image is an extension of The Quotient Image technique first introduced by Riklin-Raviv and Shashua in [11]. This method was first proposed by Wang, Li and Wang in [4].

These methods are both class recognizing methods and they are widely used for an object classifications (for example in face recognition [5]).

Original method (Quotient Image) uses series of bootstrap images to identify ideal illumination free representation of recognized class of objects. Quotient Image of two objects belonging to the same class is then defined as ratio of their albedo functions, thus being illumination free and normalizing lightning conditions and luminance variations.

Self-Quotient Image is extension of previous method that doesn't need training set of images, instead it derives Quotient Image directly from analyzed image. This means, that it can be used purely as image preprocessing method, since no direct knowledge of object's class is required.

2.3 Anisotropic diffusion

Anisotropic diffusion (also refered to as Perona-Malik diffusion) is technique first proposed by P. Perona and J. Malik in [10] that reduces noise and preserves important details of the image that are necessary for correct interpretation of the image. It is based on generating family of parametrized images, where each of these images is combination of the original image and selected filter.

3 Retinex

3.1 Introduction to Retinex

In real life huge difference in color quality of observed scene and detail of recorded image can often be perceived. The most apparent difference is loss of color accuracy and image detail, especially in darker areas covered by shadows. As a result

recorded images often seem dimmed compared to observed scene. This is caused mainly by inability of camera to distinguish between ambient illumination of the scene and reflectance. Illumination is by its nature independent of the scene, so all the characteristics of observed objects are described only by reflected light component. Recorded image is product of these two components and once it has been evaluated, there is no way we can separate these two values and obtain reflectance, which is critical for correct visual representation of the scene, but human eye still seems to be able to do so.

In 1986, Edwin Land [9] proposed image processing method that tries to simulate behavior of human eye and called it *Retinex*. Retinex is a compound of two words – Retina and Cortex - as retina and primary visual cortex are thought responsible for color constancy of final image. Since then the method has developed and can now be considered family of three main techniques:

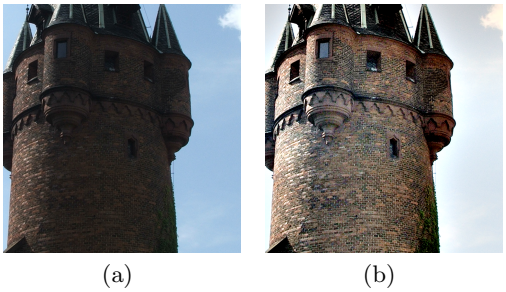
- Single Scale Retinex,
- Multi Scale Retinex,
- Multi Scale Retinex with Color Restoration (MSRCR).

3.2 Multi Scale Retinex with Color Restoration

MSRCR can be considered most advanced of these techniques and is thoroughly described in [7]. Simply put MSRCR can be described with equation:

$$R_i(x,y) = \sum_{s=1}^{N} (w_s \log I_i(x,y) - \log [F(x,y) * I_i(x,y)]) \tag{1}$$

Where i is the index of color band of the image, $R_i(x,y)$ is resulting value of pixel (x,y) of i-th color band, $I_i(x,y)$ is value of i-th color band of original image, $F(x,y)$ denotes Gaussian function and $*$ represents convolution.



(a) (b)

Fig. 1. MSRCR applied to color image. Figure (a): original image, Fig. (b): image processed with MSRCR.

Basically the MSRCR performs set of Gaussian filter operations on input image and computes difference between the filtered and unfiltered image. Each of

the steps performed is dependent on so called *scale*. Images filtered with smaller scales contain strong details and dynamic compression, but fail to provide faithful color representation. Large scales behave the opposite. The MSRCR merges all of these images and combines strengths of each scale to provide best image detail and color quality possible.

Great advantage of the MSRCR is that once desired input parameters are found technique performs constantly well with any image provided.

Example of the MSRCR applied to color image can be seen in Fig. 1. Notice how all the details (mainly bricks on the tower) became clearly visible.

4 Singular Value Decomposition

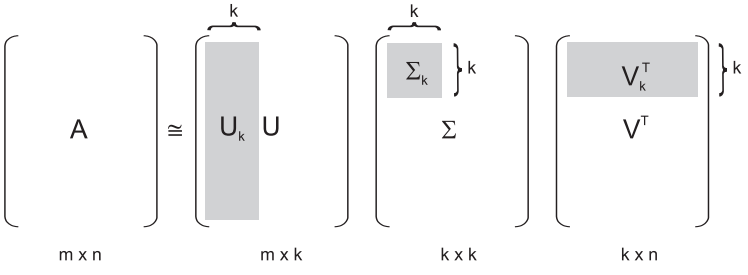


Fig. 2. k-reduced singular value decomposition

Singular value decomposition (*SVD*) is well known because of its application in information retrieval – *Latent semantic indexing* (*LSI*) [2]. It is similar to the PCA method, which has been the first method used for the generation of eigenfaces. Informally, SVD discovers significant properties and represents the images as linear combinations of the base vectors. Moreover, the base vectors are ordered according to their significance for the reconstructed image, which allows us to consider only the first k base vectors as important (the remaining ones are interpreted as "noise" and discarded). Furthermore, SVD is often referred to as more successful in recall when compared to querying whole image vectors [3].

Formally, we decompose the matrix of images A by *singular value decomposition* (*SVD*), calculating singular values and singular vectors of A.

We have matrix A, which is an $n \times m$ rank-r matrix and values $\sigma_1, \ldots, \sigma_r$ are calculated from eigenvalues of matrix AA^T as $\sigma_i = \sqrt{\lambda_i}$. Based on them, we can calculate column-orthonormal matrices $U = (u_1, \ldots, u_r)$ and $V = (v_1, \ldots, v_r)$, where $U^T U = I_n$ a $V^T V = I_m$, and a diagonal matrix $\Sigma = diag(\sigma_1, \ldots, \sigma_r)$, where $\sigma_i > 0, \sigma_i \geq \sigma_{i+1}$.

The decomposition

$$A = U \Sigma V^T$$

is called *singular decomposition* of matrix A and the numbers $\sigma_1, \ldots, \sigma_r$ are *singular values* of the matrix A. Columns of U (or V) are called *left* (or *right*) singular vectors of matrix A.

Now we have a decomposition of the original matrix of images A. We get r nonzero singular numbers, where r is the rank of the original matrix A. Because the singular values usually fall quickly, we can take only k greatest singular values with the corresponding singular vector coordinates and create a *k-reduced singular decomposition* of A.

Let us have k $(0 < k < r)$ and singular value decomposition of A

$$A = U\Sigma V^T \approx A_k = (U_k U_0) \begin{pmatrix} \Sigma_k & 0 \\ 0 & \Sigma_0 \end{pmatrix} \begin{pmatrix} V_k^T \\ V_0^T \end{pmatrix}$$

We call $A_k = U_k \Sigma_k V_k^T$ a k-reduced singular value decomposition (rank-k SVD) (U_0, Σ_0, and V_0 represent matrices filled with zeros).

Instead of the A_k matrix, a matrix of image vectors in reduced space $D_k = \Sigma_k V_k^T$ is used in SVD as the representation of image collection. The image vectors (columns in D_k) are now represented as points in k-dimensional space (the *feature-space*). For an illustration of rank-k SVD see Figure 2.

Rank-k SVD is the best rank-k approximation of the original matrix A. This means that any other decomposition will increase the approximation error, calculated as a sum of squares (*Frobenius norm*) of error matrix $B = A - A_k$. However, it does not implicate that we could not obtain better precision and recall values with a different approximation.

To execute a query Q in the reduced space, we create a reduced query vector $q_k = U_k^T q$ (another approach is to use a matrix $D'_k = V_k^T$ instead of D_k, and $q'_k = \Sigma_k^{-1} U_k^T q$). Instead of A against q, the matrix D_k against q_k (or q'_k) is evaluated.

Once computed, SVD reflects only the decomposition of original matrix of images. If several hundreds of images have to be added to existing decomposition (*folding-in*), the decomposition may become inaccurate. Because the recalculation of SVD is expensive, so it is impossible to recalculate SVD every time images are inserted. The *SVD-Updating* [3] is a partial solution, but since the error slightly increases with inserted images. If the updates happen frequently, the recalculation of SVD may be needed soon or later.

5 Detecting frayed edges on grain oriented electrical steel

The Retinex as the most appropriate image normalization technique was chosen for preprocessing of images in inspection of quality in grain oriented electrical steel making process.

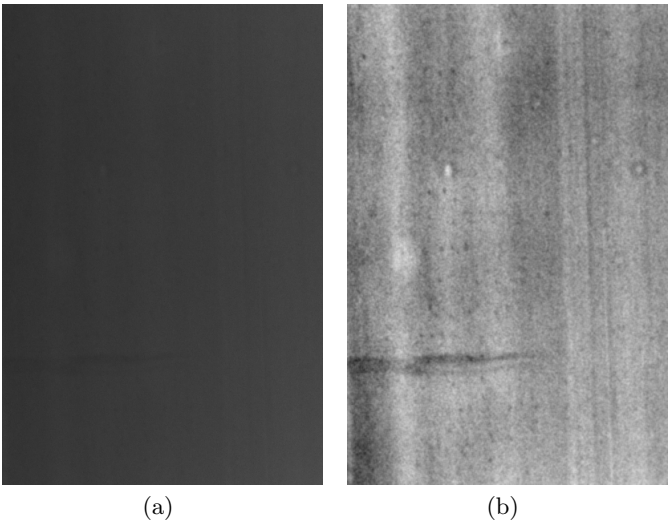
Frayed edges detection is part of surface quality monitoring system. Goal of the system is to monitor grain oriented electrical steel plate's surface during manufacturing process and to detect set of defects degrading quality of final product.

Steel plate is coiled up into the coils. Approximate length of one coil is 4000 meters.

Steel plate continuously runs through the de-carbonization line and it's surface is monitored by set of cameras from both sides. System then analyses input images for defects in real-time.

One of the most problematic defects to detect is frayed edge. In the input image it appears only as a small deviation in brightness in horizontal direction (see Fig. 3). This type of defect is captured from one side of the plate only (as it is visible from both sides) using monochrome digital camera. Resolution of one image is 2400×600 pixels. Width of area captured by one camera is approximately 0.5 m, which means that each millimeter of captured area is represented almost by 5 pixels in final image. Images are automatically archived so they can be worked with to improve quality of defect detecting algorithms and now we currently have base of more then two million test images.

Frayed edges arise on a plate because of insufficient MgO powder coverage of the edges. In the annealing process the uncovered edges are stuck together, because of high annealing temperature, and the defect is formed on a plate when it is unwinded on the next processing line and stuck edges are torn off.



(a) (b)

Fig. 3. Example of frayed edge. On Fig. (a) we can see original image, on Fig. (b) is the same image processed with Retinex.

Common edge detecting algorithms used on non-preprocessed images do not provide any meaningful results because frayed edge appears only as very small deviation in input image and is suppressed by noise that is introduced into the image due to low exposition time requirements and environmental conditions that do not allow for better lighting of the scene.

To highlight our area of interest – the frayed edge – and to suppress the light non-constancy is the core of the problem.

Many preprocessing algorithms were tried before the Retinex was chosen for this problem. Among others these light normalization algorithms where tried out: Histogram Normalization [1], Self Quotient Image [4], Anisotropic Diffusion [10] and many more.

6 Experiment Design

In our experiments we will try to detect frayed edges on plate using simple Sobel filter [8] that will be applied after all preprocessing algorithms. Results of this Sobel filter and some additional processing (filtering edges caused by noise ...) will allow us to judge quality of used preprocessing algorithms.

SVD's result will be used as a mask on preprocessed image. This will allow us to perform detection only on areas highlighted by the SVD. First we will try to apply this mask to original input image, so we can see if the SVD itself is able to replace the Retinex (if the SVD's mask is accurate enough then the Sobel filter might yield correct results). Additionally we will try to apply mask to image already preprocessed with the Retinex to see whether it can improve accuracy of the Sobel filter. Finally we will run the Sobel filter on image processed purely the Retinex for reference.

SVD image can be computed either from input image or from image already processed by the Retinex algorithm. Both variants will be tried in our experiments.

As test database life data captured with system described in Sect. 5 will be used. Database consists of 100 images containing Frayed Edge on left edge of the steel plate.

All relevant parameters of preprocessing algorithms used for experiments are summarized in Table 1. First column *Name* of the table contains identification name of the run. Second column *SVD source* specifies whether the SVD was computed from original image or from image with the Retinex applied. Third column *Source image* specifies whether the SVD's mask will be used on original image or on image with the Retinex. Column *Multiplier* contains value that will be used to multiply all values in resulting image to enhance the brightness of the image. Column *Lower bound* contains lower bound constant, all values lower than this bound will be clipped to 0. Last column *Upper bound* contains upper bound constant, all values higher then this constant will be automatically adjusted to 255. First row of the table represents reference algorithm run.

First we will run our reference algorithm and store locations and depths of frayed edges present on the image. Then all the other settings will be run and their results will be compared to those of the reference run.

To conclude contributions of SVD to the detection the following metric will be used:

- if the Sobel filter is able to detect at least 80 % of frayed edge's length, then the detection is considered as successful,

Table 1. Parameters of experiments.

Name	SVD source	Source image	Multiplier	Lower bound	Upper bound
Retinex	-	-	-	-	-
SVD 1	Original image	Original image	100	200	200
SVD 2	Original image	Retinex image	100	200	200
SVD 3	Retinex image	Original image	100	200	200
SVD 4	Retinex image	Retinex image	100	200	200
SVD 5	Original image	Original image	100	150	225
SVD 6	Original image	Retinex image	100	150	225
SVD 7	Retinex image	Original image	100	150	225
SVD 8	Retinex image	Retinex image	100	150	225

- if the algorithm detects false frayed edge of length at least 10 % of the image's size, then it is considered as false positive,
- otherwise detection is considered unsuccessful.

Ratio of successful detections and sum of false positives and unsuccessful detections will then be our metric that will allow us to compare individual settings. This metric is described in Eq. (2).

$$m = \frac{s}{s + f + u} \cdot 100\ \% \tag{2}$$

Where:

- m - final number specifying accuracy in percents of given run, the higher the number the better,
- s - number of successfully detected images,
- f - number of false positive detections,
- u - number of unsuccessful detections.

7 Experiment Results

Table 2 summarizes results we have obtained. Structure of the table is similar to Table 1 only column *Success rate* is added. This column contains result of the experiment, how this number was obtained is described in Sect. 6 and in Eq. (2).

We can see that the SVD by itself was not able to highlight defected areas sufficiently. All runs that were detecting the defect from original image failed to detect single defected image from the testing set.

Runs that used the Retinex image as source image for detection were able to detect defects with some success. Those that used the Retinex as source both for the SVD and for detection performed much better. Those that used the Retinex only as source image and SVD mask was computed from original image performed unconvincingly. This is caused by fact that SVD highlighted only

Table 2. Experiment results.

Name	SVD source	Source image	Multiplier	Lower bound	Upper bound	Success rate
Retinex	-	-	-	-	-	100 %
SVD 1	Original image	Original image	100	200	200	0 %
SVD 2	Retinex image	Original image	100	200	200	0 %
SVD 3	Original image	Retinex image	100	200	200	16 %
SVD 4	Retinex image	Retinex image	100	200	200	91 %
SVD 5	Original image	Original image	100	150	225	0 %
SVD 6	Retinex image	Original image	100	150	225	0 %
SVD 7	Original image	Retinex image	100	150	225	42 %
SVD 8	Retinex image	Retinex image	100	150	225	91 %

significantly different areas in the image and omitted the less distinctive ones thus shortening the length of detected defect.

Sample images of all settings can be found in Fig. 4. We can see original image 4(a), Retinex image 4(b), SVD (upper bound: 200, lower bound: 200) computed from original image 4(c), SVD(upper bound: 200, lower bound: 200) computed from the Retinex image 4(d), SVD (upper bound: 225, lower bound: 125) computed from original image 4(e) and SVD(upper bound: 225, lower bound: 125) computed from Retinex image 4(f).

From the results obtained we can see that the SVD itself led to no improvements in detection. As an algorithm to highlight defected areas the SVD itself failed and even with help of the Retinex algorithm it was not able to achieve 100 % success rate, so the addition of the SVD will not be an improvement over the current the Retinex solution and will not help to detect defects that the Retinex previously was not able to.

8 Conclusion

In this paper we have tried to use Singular Value Decomposition to improve accuracy of frayed edge detection. Proposition that the SVD itself might be able to successfully highlight defected areas has proven to be wrong as it was not able to correctly detect single frayed edge in test images.

When used in combination with the Retinex the detection algorithm was able to achieve 91 % success rate of reference the Retinex algorithm. This means, that with addition of this mask the algorithm performed worse then without it. Our hopes were that with usage of the SVD mask the detection algorithm will detect defects in all test images and we might be able to lower the threshold on edge detection algorithm thus finding more subtle frayed edges and improving the accuracy of the algorithm. With success rate of 91 % this idea is proven to be wrong.

To summarize results presented in this paper – method introduced in our previous paper [6] still achieves best results we were able to obtain so far. Usage

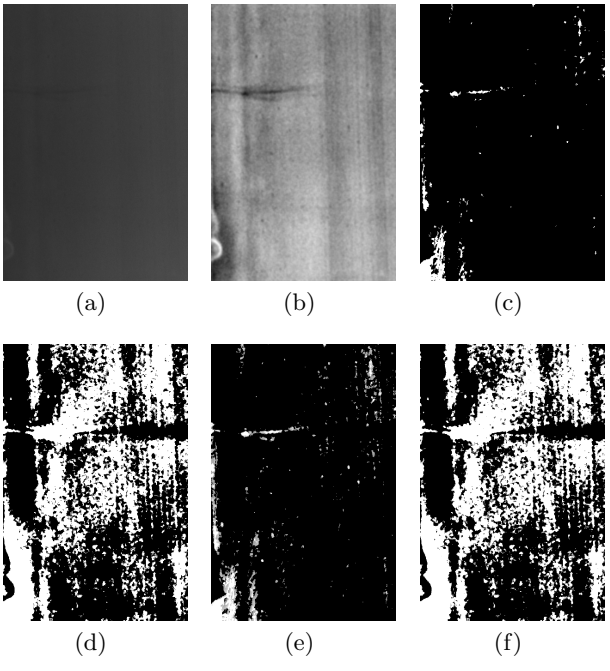


Fig. 4. (a) - Original, (b) - Retinex, (c) - SVD1, (d) - SVD3, (e) - SVD5, (f) - SVD7

of the SVD both as a pure defect detection algorithm or as a mask only lowers success rate of the detection and makes usage of the SVD in our algorithm pointless.

In our future work we would like to test more lighting normalization methods and try to improve detection accuracy so that the algorithm would be able to reliably detect even more subtle frayed edges.

Acknowledgment

This work was supported by the European Regional Development Fund in the IT4Innovations Centre of Excellence project (CZ.1.05/1.1.00/02.0070) and by the Development of human resources in research and development of latest soft computing methods and their application in practice project, reg. no. CZ.1.07/2.3.00/20.0072 funded by Operational Programme Education for Competitiveness, co-financed by ESF and state budget of the Czech Republic.

References

1. Acharya and Ray. Image processing: Principles and applications. *Wiley-Interscience*, 2005.

2. M.W. Berry and M. Browne. *Understanding Search Engines, Mathematical Modeling and Text Retrieval*. Siam, 1999.
3. M.W. Berry, S.T. Dumais, and T.A. Letsche. Computation Methods for Intelligent Information Access. In *Proceedings of the 1995 ACM/IEEE Supercomputing Conference*, 1995.
4. S. Li H. Wang and Y. Wang. Face recognition under varying lighting conditions using self quotient image. *IEEE International Conference on Automatic Face and Gesture Recognition*, 2004.
5. Guillaume Heusch, Fabien Cardinaux, and Sebastien Marcel. Lighting normalization algorithms for face verification. 2005.
6. Michal Holis and Martin Placek. Detecting frayed edge in steel plate using multi scale retinex algorithm. In *Wofex*, pages 440–446. VŠB TUO, 2012.
7. D.J. Jobson, Z. Rahman, and G.A. Woodell. A multiscale retinex for bridging the gap between color images and the human observation of scenes. *IEEE Transactions on Image Processing*, 6:965 – 976, 1997.
8. N. Kanopoulos. Design of an image edge detection filter using the sobel operator. *Solid-State Circuits, IEEE Journal of*, 23:358 – 367, 1988.
9. E. Land. An alternative technique for the computation of the designator in the retinex theory of color vision. *Proc. Nat. Acad*, 83:3078–3080, 1986.
10. P. Perona and J. Malik. Scale-space and edge detection using anisotropic diffusion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(7), 1990.
11. A. Shashua and T. Riklin-Raviv. The quotient image : Class based re-rendering and recognition with varying illuminations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(2), 2001.

Application of Relative Derivation Terms by Polynomial Neural Networks

Ladislav Zjavka

VŠB – Technical University of Ostrava, IT4innovations Ostrava, Czech Republic
ladislav.zjavka@vsb.cz

Abstract. A lot of problems involve unknown data relations, which can define a derivative based model of dependent variables generalization. Standard soft-computing methods (as artificial neural networks or fuzzy rules) apply usual absolute interval values of input variables. The new proposed differential polynomial neural network makes use of relative data, which can better describe the character regarding a wider range of input values. It constructs and resolves an unknown partial differential equation, using fractional polynomial sum derivative terms of relative data changes. This method might be applied to solve problems concerned a visual pattern generalization or complex system modeling.

1 Introduction

Differential equations are able to solve a variety of pattern recognition and function approximation problems [2]. A principal lack of the artificial neural network (ANN) behavior in general is a disability of the data relation generalization [8]. Differential polynomial neural network (D-PNN) is a new neural network type, designed by the author, which creates and resolves an unknown partial differential equation (DE) of a multi-parametric function approximation. A DE is replaced producing sum of fractional polynomial derivative terms, forming a system model of dependent variables. Its regression is not based on a simple whole-pattern affinity but learned generalized data relations. This seems to be mainly profitable by application of different learning and testing interval values of input variables. Standard soft-computing methods usual are not able to operate correctly on varying training and testing data range, utilizing only the absolute values.

$$y = a_0 + \sum_{i=1}^{m} a_i x_i + \sum_{i=1}^{m}\sum_{j=1}^{m} a_{ij} x_i x_j + \sum_{i=1}^{m}\sum_{j=1}^{m}\sum_{k=1}^{m} a_{ijk} x_i x_j x_k + \ldots \quad \textbf{(1)}$$

m – number of variables
$A(a_1, a_2, \ldots, a_m)$, ... - vectors of parameters $X(x_1, x_2, \ldots, x_m)$ - input vector

D-PNN resulted from the GMDH polynomial neural network (Fig.1.), which was created by a Ukrainian scientist Aleksey Ivakhnenko in 1968 [3]. When the back-

© V. Snášel, K. Richta, J. Pokorný (Eds.): Dateso 2013, pp. 98–108, ISBN 978-80-248-2968-5.
VŠB – Technical University of Ostrava, FEECS, Department of Computer Science, 2013.

propagation technique was not known yet a technique called Group Method of Data Handling (GMDH) was developed for neural network structure design and parameters of polynomials adjustment. General connection between input and output variables is expressed by the Volterra functional series, a discrete analogue of which is Kolmogorov-Gabor polynomial (1). This polynomial can approximate any stationary random sequence of observations and can be computed by either adaptive methods or system of Gaussian normal equations.

$$y' = a_0 + a_1 x_i + a_2 x_j + a_3 x_i x_j + a_4 x_i^2 + a_5 x_j^2 \qquad \textbf{(2)}$$

GMDH decomposes the complexity of a process into many simpler relationships each described by low order polynomials (2) for every pair of the input values. Typical GMDH network maps a vector input $\boldsymbol{x}$ to a scalar output y', which is an estimate of the true function $f(x) = y$. Each neuron of the polynomial network fits its output to the desired value y for each input vector $\boldsymbol{x}$ from the training set. It defines an optimal structure of complex system model with identifying non-linear relations between input and output variables [5].

2 Differential polynomial neural network

The basic idea of the D-PNN is to create and replace a partial differential equation (DE) (3), which is not known in advance and is able to describe a system of dependent variables, with a sum of fractional multi-parametric polynomial derivative terms (4)[2].

$$a + \sum_{i=1}^{n} b_i \frac{\partial u}{\partial x_i} + \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} \frac{\partial^2 u}{\partial x_i \partial x_j} + \ldots = 0 \qquad u = \sum_{k=1}^{\infty} u_k \qquad \textbf{(3)(4)}$$

$u(x_1, x_2, \ldots, x_n)$ - searched function of all input variables (dependent variable)
a, $B(b_1, b_2, \ldots, b_n)$, $C(c_{11}, c_{12}, \ldots)$ - parameters

The applied method of integral analogues replaces math operators and symbols of a DE by ratio of corresponding variables. Derivatives are replaced by the integral analogues, i.e. derivative and all operators are replaced by analogous or proportion marks in equations [4].

$$u_i = \frac{\left(a_0 + a_1 x_1 + a_2 x_2 + a_3 x_1 x_2 + a_4 x_1^2 + a_5 x_2^2 + \ldots\right)^{m/n}}{b_0 + b_1 x_1 + \ldots} = \frac{\partial^m f(x_1, \ldots, x_n)}{\partial x_1 \partial x_2 \ldots \partial x_m} \qquad \textbf{(5)}$$

n – combination degree of a complete polynomial of n-variables
m – combination degree of denominator

The numerator of a term (5) is a polynomial of all n-input variables of a single neuron and partly defines an unknown function u of eq. (3)(4). The denominator is a derivative part of a DE term (5), which arose from the partial derivation of the complete n-variable polynomial by competent variable(s). The root function of numerator takes the polynomial into competent combination degree but needn't be used at all if not necessary.

A block of the D-PNN (Fig.1.) consists of basic neurons, one for each fractional polynomial (5), defining a sum partial derivative term of the DE (3) solution. Blocks of higher layers are additionally extended with compound neurons of composite functions, which apply previous layer block outputs and inputs. Each block contains a single output polynomial (without derivative part), thus the block skeleton of the D-PNN is formed by the GMDH network. Neurons don't affect the block output but are applied directly in the sum of a total output calculation of a PDE composition (4). Each block has 1 and neuron 2 vectors of adjustable parameters *a*, resp. *a*, *b*.

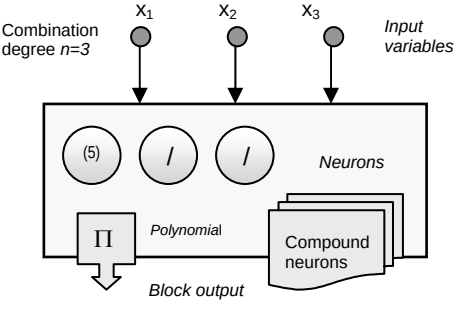


Fig. 1. D-PNN block of basic and compound neurons

Root mean square (RMS) error method (6) was applied for polynomial parameter optimization and PDE term selection.

$$E = \sqrt{\frac{\sum_{i=1}^{M} \left(y^d - y_i\right)^2}{M}} \rightarrow \min \qquad (6)$$

3 Multi-layered backward D-PNN

Multi-layered D-PNN forms composite polynomial functions (Fig.2.). Compound DE terms, i.e. derivatives in respect to variables of previous layers, are calculated according to the composite function partial derivation rules (7)(8). They are formed by products of partial derivatives of external and internal functions.

$$F(x_1, x_2, \dots, x_n) = f(y_1, y_2, \dots, y_m) = f(\phi_1(X), \phi_2(X), \dots, \phi_m(X)) \qquad i = 1, \dots, m \qquad (7)$$

$$\frac{\partial F}{\partial x_k} = \sum_{i=1}^{m} \frac{\partial f(y_1, y_2, \dots, y_m)}{\partial y_i} \cdot \frac{\partial \phi_i(X)}{\partial x_k} \qquad k=1, \dots, n \qquad (8)$$

Each block of the D-PNN involves basic neurons e.g. (9), at first of only linear regression. Additionally blocks of the 2nd and following hidden layers are also extended with neurons, which form composite derivatives utilizing outputs and inputs

of back connected previous layer blocks, e.g. the 1st block of the last (3rd) hidden layer (10)(11) [7].

$$y_1 = \frac{\partial f(x_{21}, x_{22})}{\partial x_{21}} = w_1 \frac{(a_0 + a_1 x_{21} + a_2 x_{22} + a_3 x_{21} x_{22})^{1/2}}{2 \cdot (b_0 + b_1 x_{21})} \quad \textbf{(9)}$$

$$y_2 = \frac{\partial f(x_{21}, x_{22})}{\partial x_{11}} = w_2 \frac{(a_0 + a_1 x_{21} + a_2 x_{22} + a_3 x_{21} x_{22})^{1/2}}{2 \cdot x_{22}} \cdot \frac{(x_{21})^{1/2}}{2 \cdot (b_0 + b_1 x_{11})} \quad \textbf{(10)}$$

$$y_3 = \frac{\partial f(x_{21}, x_{22})}{\partial x_1} = w_3 \frac{(a_0 + a_1 x_{21} + a_2 x_{22} + a_3 x_{21} x_{22})^{1/2}}{2 \cdot x_{22}} \cdot \frac{(x_{21})^{1/2}}{2 \cdot x_{12}} \cdot \frac{(x_{11})^{1/2}}{2 \cdot (b_0 + b_1 x_1)} \quad \textbf{(11)}$$

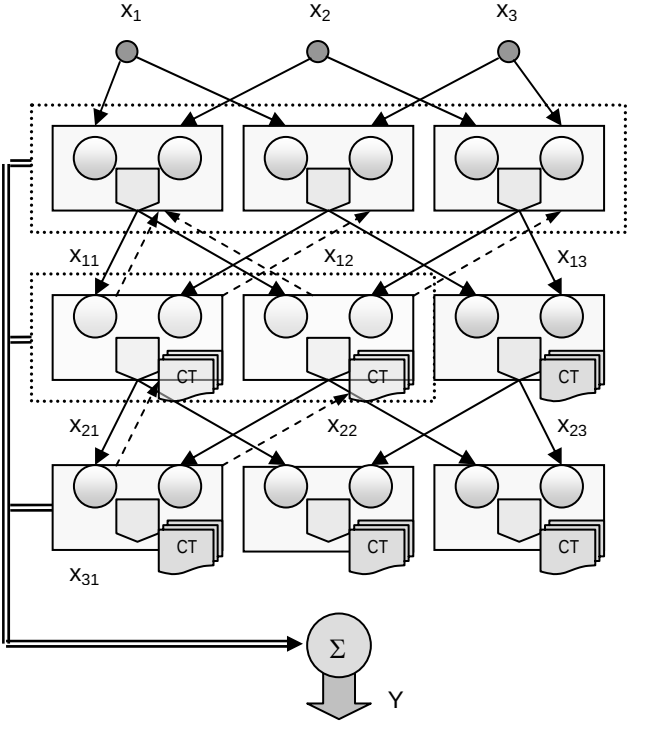


Fig. 2. 3-variable 2-combination block D-PNN

The best-fit neuron selection is the initial phase of the DE composition and may apply a proper genetic algorithm (GA). Parameters of polynomials might be adjusted by means of difference evolution algorithm (EA), supplied with sufficient random mutations [1]. The parameter optimization is performed simultaneously with the GA term combination search, where may arise a quantity of local and global error solutions. There would be welcome to apply an adequate gradient descent method

too, which parameter updates result from partial derivatives of polynomial DE terms in respect with the single parameters [6]. The number of network hidden layers coincides with a total amount of input variables.

$$Y = \frac{\sum_{i=1}^{k} y_i}{k} \qquad k = \textit{amount of active DE terms} \tag{11}$$

Only some of all potential combination DE terms (neurons) may participate in the DE composition, in despite of they have an adjustable term weight (w_i). D-PNN's total output Y is the sum of all active neuron outputs, divided by their amount k (11).

4 Identification of data relations

Consider first only a linear simplification of data relations, thus only linear polynomials of neurons and blocks might be applied. D-PNN consisting of only 1 block of 2 neurons, all terms of the DE (12), is able to identify simple linear 2-variable dependence (function), e.g. $x_1 = 2x_2$.

$$y = w_1 \frac{\left(a_0 + a_1x_1 + a_2x_2 + a_3x_1x_2\right)^{1/2}}{b_0 + b_1x_1} + w_2 \frac{\left(a_0 + a_1x_1 + a_2x_2 + a_3x_1x_2\right)^{1/2}}{b_0 + b_1x_2} \tag{12}$$

More complicated dependence, where 2 variables depend on a 3rd (e.g. $x_1 + x_2 = x_3$) may be resolved again D-PNN with one 3-variable combination block. The complete DE (of 1 and 2-combination derivatives) consists of 6 sum terms (neurons) but only 3 may be employed, derivative terms for x_3 (13), x_1x_3 (14), x_2x_3 (15). Some neurons must be inactivated, having an undesirable effect on the network correct operation. The applied 2-variable combination block D-PNN has 3 hidden layers (Fig.2.).

$$y_1 = w_1 \frac{\left(a_0 + a_1x_1 + a_2x_2 + a_3x_3 + a_4x_1x_2 + \ldots + a_7x_1x_2x_3\right)^{1/3}}{b_0 + b_1x_3} \tag{13}$$

$$y_2 = w_2 \frac{\left(a_0 + a_1x_1 + a_2x_2 + a_3x_3 + a_4x_1x_2 + \ldots + a_7x_1x_2x_3\right)^{2/3}}{b_0 + b_1x_1 + b_2x_3 + b_3x_1x_3} \tag{14}$$

$$y_3 = w_3 \frac{\left(a_0 + a_1x_1 + a_2x_2 + a_3x_3 + a_4x_1x_2 + \ldots + a_7x_1x_2x_3\right)^{2/3}}{b_0 + b_1x_2 + b_2x_3 + b_3x_2x_3} \tag{15}$$

D-PNN can indicate the learned dependence of 3 variables (function) by the output value *1.0* (or any desired). It was trained with only 6 data samples (Tab.1), which were selected to involve proportionally the whole training data interval values *<0,500>*. However the output function values $x_3=x_1+x_2$ (*x*-axis) of the test random input vectors can exceed the maximal trained sum value *500*, while the response is

kept (Fig.3.). Output errors can result from very disproportional random vector values, which D-PNN was not trained to, e. g. *360 = 358 + 2*.

Table 1. Training data set of the 3-variable dependence (function) identification $x_1 + x_2 = x_3$

	1	2	3	4	5	6
x_1	1	70	40	160	30	300
x_2	2	3	100	60	330	200
x_3	3	73	140	220	360	500

The identification of data relations might be applied to a generalization of fragmented visual patterns into some characteristic dependent elements, which shape assume moved or sized form in the input matrix and where ANN applications fail [7]. The outcomes of 1-block and multi-layered D-PNN are comparable, however the 2nd type is able to involve far larger amount of DE terms and so form a more accurately description of a model.

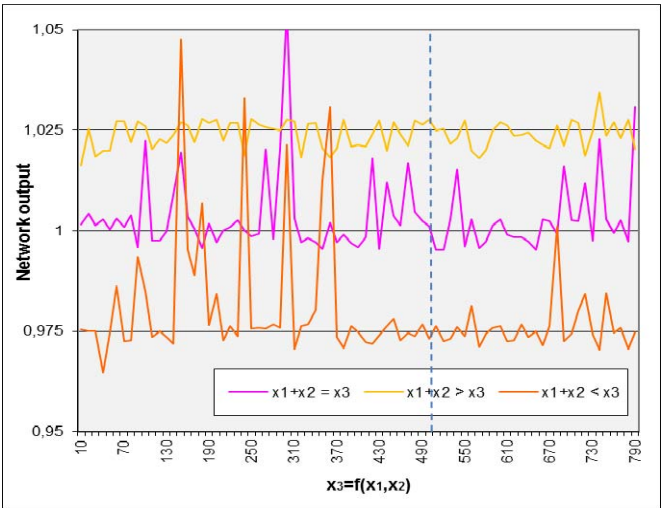


Fig. 3. Identification of a multi-parametric function relation

5 Function approximations

D-PNN can approximate a multi-parametric function, analogously to the ANN approach. Consider the sum function again $y^t = x_1 + x_2 + x_3$, however it could be any linear function. The network with 3 input variables, forming 1 output $y = f(x_1, x_2, x_3)$ should approximate the true function y^t by means of sum derivative terms of the partial DE solution. The training data was necessary to be doubled into 12 samples (Tab.2). The D-PNN and ANN approximation is co-equal on the trained interval values $<6, 520>$, however the ANN approximation ability rapidly falls outside of this range (Fig.4.). The type and operating principle of the D-PNN is the same with

applied the dependence identification (Fig.2.), though requiring more time-consuming adjustment.

Table 2. The $y^t = x_1 + x_2 + x_3$ function approximation training data set

	1	**2**	**3**	**4**	**5**	**6**
$\mathbf{x_1}$	1	70	4	160	200	30
$\mathbf{x_2}$	2	3	100	90	20	330
$\mathbf{x_3}$	3	20	40	10	100	20
$\mathbf{y^d}$	6	93	144	260	320	380

	7	**8**	**9**	**10**	**11**	**12**
$\mathbf{x_1}$	4	10	150	20	50	260
$\mathbf{x_2}$	5	70	5	210	150	60
$\mathbf{x_3}$	12	80	55	100	200	200
$\mathbf{y^d}$	21	160	210	330	400	520

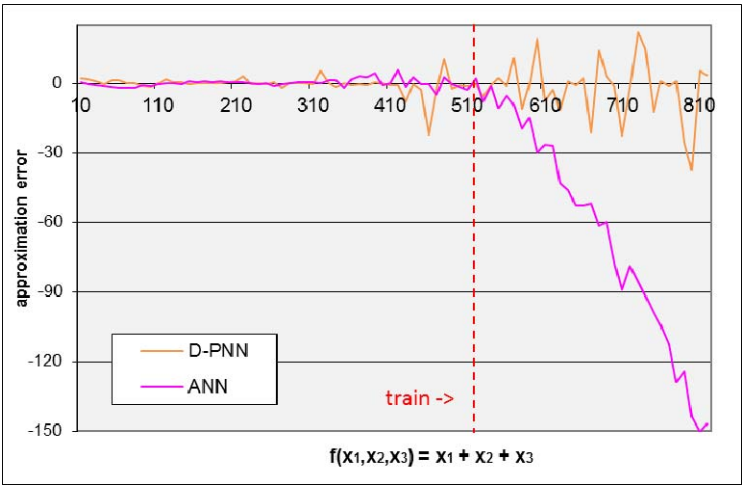


Fig. 4. Comparison of a linear multi-parametric function approximation

$$F\left(x, y, u, \frac{\partial u}{\partial x}, \frac{\partial u}{\partial y}, \frac{\partial^2 u}{\partial x^2}, \frac{\partial^2 u}{\partial x \partial y}, \frac{\partial^2 u}{\partial y^2}\right) = 0 \qquad \textbf{(16)}$$

where F(x, y, u, p, q, r, s, t) is a function of 8 variables

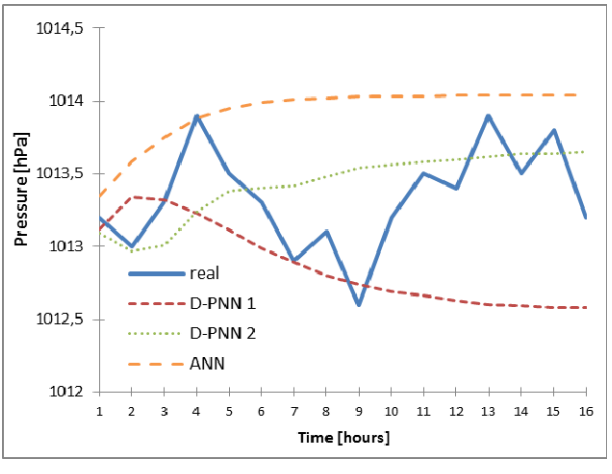
In the case of a real-data application D-PNN processes 2-combination square polynomials of blocks and neurons (DE terms), the same as applied by the GMDH algorithm (2). This simple polynomial type proves to yield best results besides an easy use and improves also the linear function approximation (which is notable). Thus each block includes 5 basic neurons of derivatives x_1, x_2, x_1x_2, x_1^2, x_2^2 of the 2nd

order partial DE (3) of an unknown 2-variable function *u*, which might be transferred into form of eq. (16). Without this extension only a linear regression of the training data set would be applied. The square and combination derivative terms are also calculated according to the composite function derivation rules (17)(18). However they don't apply the complete sum of the formulas but only 2 simple terms with 1st order external function derivatives e.g. (19).

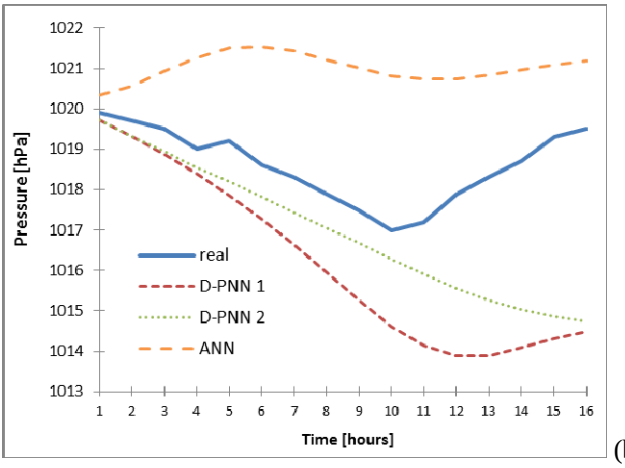
$$F(x,y) = f(u,v) = f[\varphi(x,y),\psi(x,y)] \tag{17}$$

$$\frac{\partial^2 F}{\partial x^2} = \frac{\partial^2 f}{\partial u^2}\left(\frac{\partial \varphi}{\partial x}\right)^2 + 2\frac{\partial^2 f}{\partial u \partial v}\frac{\partial \varphi}{\partial x}\cdot\frac{\partial \psi}{\partial x} + \frac{\partial^2 f}{\partial v^2}\left(\frac{\partial \psi}{\partial x}\right)^2 + \frac{\partial f}{\partial u}\cdot\frac{\partial^2 \varphi}{\partial x^2} + \frac{\partial f}{\partial v}\cdot\frac{\partial^2 \psi}{\partial x^2} \tag{18}$$

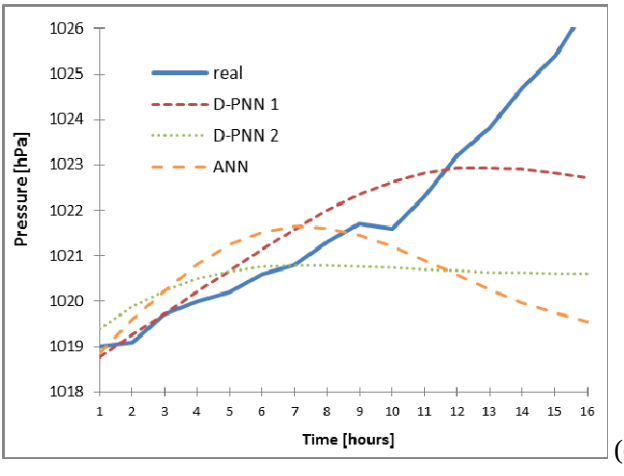
$$y_4 = \frac{\partial^2 f(x_{21},x_{22})}{\partial x_{11}^2} = w_4 \frac{(a_0 + a_1 x_{21} + a_2 x_{22} + a_3 x_{21} x_{22} + a_4 x_{21}^2 + a_5 x_{22}^2)^{1/2}}{3 \cdot x_{22}} \cdot \frac{x_{21}}{2 \cdot (b_0 + b_1 x_{11} + b_2 x_{11}^2)} \tag{19}$$



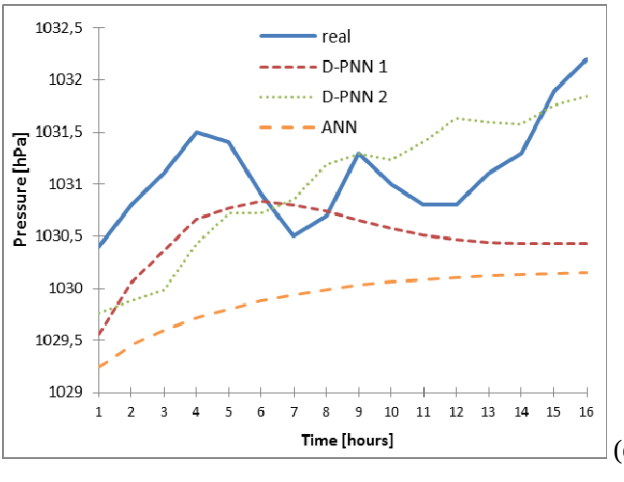
(a)


Pressure [hPa]
Time [hours]
real
D-PNN 1
D-PNN 2
ANN

(b)


Pressure [hPa]
Time [hours]
real
D-PNN 1
D-PNN 2
ANN

(c)


Pressure [hPa]
Time [hours]
real
D-PNN 1
D-PNN 2
ANN

(d)

Fig. 5a-d. Comparison of static pressure time-series predictions

The 3-variable D-PNN applying extended polynomials (2) of blocks, neurons and square, combination DE terms, tried to predict the static pressure of 1 site locality time-series (Fig.5a-d). It was trained along with the 1-layer ANN the previous day hourly pressure data series (24 or 48 hours, i.e. data samples), which are free on-line available [9]. Meteorological forecasts require as a rule high amount of state input variables to define a complex model, however some tendencies of the progress curves are notable. The D-PNN 2 models double the applied amount of neurons compared with D-PNN 1 (Fig.5.). The DE composition based predictions of the D-PNN seems to succeed any better. The more varied models are formed than ANN (applying 4 or 5 input variables), which is induced by a different neuron combination selection.

6 Conclusion

D-PNN is a new neural network type, which identification and function approximation is based on generalization of data relations. The relative data processing is contrary to common soft-computing method approachs (e.g. ANN), which applications are subjected to a fixed interval of absolute values. This handicap disallows to use various learning and testing data range values (Fig.4.), which may involve real data applications. Thus D-PNN's non-linear regression can cover a generalization of wider interval values. It forms and resolves a DE, composed of sum fractional derivative terms, defining a system model of dependent variables. It is trained only with a small set of input-output data samples, likewise the GMDH algorithm does [1]. The inaccuracies of presented experiments can result from applied incomplete rough training and selective methods, requiring large improvements. Behavior of the presented method differs essentially from other common neural network techniques.

Acknowledgement

This work has been elaborated in the framework of the IT4Innovations Centre of Excellence project, reg. no. CZ.1.05/1.1.00/02.0070 supported by Operational Programme 'Research and Development for Innovations' funded by Structural Funds of the European Union and by the Ministry of Industry and Trade of the Czech Republic, under the grant no. FR-TI1/420 , and by SGS, VŠB – Technical University of Ostrava, Czech Republic, under the grant No. SP2012/58.

References

1. Das, S., Abraham, A., Konar, A.: Particle swarm optimization and Differential evolution algorithms. Studies in Computational Intelligence (SCI) 116, 1-38, 2008. Springer-Verlag Berlin.
2. Iba, H.: Inference of differential equation models by genetic programming. Information Sciences, Volume 178, Issue 23, 1 December 2008, Pages 4453–4468.
3. Ivakhnenko, A.G.: Polynomial theory of complex systems. IEEE Transactions on systems, Vol. SMC-1, No.4. 1971.
4. Kuneš, J., Vavroch, O., Franta, V.: Essentials of modeling. SNTL Praha 1989 (in Czech).
5. Nikolaev, N.Y., Iba, H.: Adaptive Learning of Polynomial Networks. Springer, New York 2006.
6. Nikolaev, N. Y., Iba, H.: Polynomial harmonic GMDH learning networks for time series modelling. Neural Networks 16 (2003), 1527–1540. Science Direct.
7. Zjavka, L. : Generalization of patterns by identification with polynomial neural network. Journal of Electrical Engineering Vol. 61, No. 2/2010, p. 120-124
8. Zjavka, L.: Recognition of Generalized Patterns by a Differential Polynomial Neural Network. Engineering, Technology & Applied Science Research Vol. 2, No 1 (2012).
9. National Climatic Data Center of National Oceanic and Atmospheric Administration (NOAA) http://cdo.ncdc.noaa.gov/cdo/3505dat.txt

Evolution of Co-Authors Communities Formed by Terms on DBLP

Alisa Babskova, Pavla Dráždilová, Jan Martinovič, Václav Svatoň, and Václav Snášel

VŠB – Technical University of Ostrava
Faculty of Electrical Engineering and Computer Science
17. listopadu 15/2172, 708 33 Ostrava, Czech Republic
{alisa.babskova.st,pavla.drazdilova,jan.martinovic}@vsb.cz
{vaclav.svaton.st,vaclav.snasel}@vsb.cz

Abstract. The DBLP Computer Science Bibliography server provides bibliographic information on major computer science journals and proceedings. DBLP indexes more than 2.1 million articles and contains titles of articles, their authors, years of publication etc. Downloadable DBLP dataset is very interesting resource for evolution analysis of co-author networks. The paper deals with subgraphs of the authors from DBLP with common interests. The common interest of the authors is defined by terms, which are extracted from the titles of articles. The subgraphs are extracted for each year separately based on the published years. These subgraphs represent the communities of co-authors, for which is observed their development in time. That new view of these communities of the co-authors offer a new way for analysis and measurement of article datasets.

1 Introduction

The aim of this paper was to develop a methodology for finding, tracking, analysing and evaluating the development of the groups of authors who deal with the areas specified by chosen terms. We can see whether this area is still developing, expires, is stable or promising. The results of this paper could be used by researchers to point their professional interest.

Our work has been inspired by papers in which the authors tried to analyse dynamic aspects of communities. Authors present in the paper [5] a framework for modelling and detecting community evolution over time. They proposed the community matching algorithm which efficiently identifies and tracks similar communities over time. A series of significant events and transitions is defined to characterize the evolution of networks in the terms of its communities and individuals. The authors also propose two metrics called stability and influence metrics to describe the active behaviour of the individuals. They present experiments to explore the dynamics of communities on the Enron email and DBLP datasets.

In the paper [14] authors construct word association network from DBLP bibliography records based on word concurrence relationship in titles and analyse statistical distribution of edge frequency. The authors find that frequency distribution of the word also satisfy power-law distribution.

© V. Snášel, K. Richta, J. Pokorný (Eds.): Dateso 2013, pp. 109–118, ISBN 978-80-248-2968-5.
VŠB – Technical University of Ostrava, FEECS, Department of Computer Science, 2013.

The paper [3] was written to address the question which communities will grow rapidly, and how do the overlaps among the pairs of communities change over time. In the paper were used two large sources of data: friendship links and community membership on LiveJournal, and co-authorship and conference publications in DBLP. Authors of this work studied how the evolution of these communities relates to properties such as the structure of the underlying social networks.

In the article [8] authors show an interesting metrics for evaluating communities evolving in time. For their experiment they consider data sets of the monthly list of articles in the Cornell University Library e-print condensed matter archive and the record of phone calls between the customers of a mobile phonecompany. About this metrics we will talk more in Section 4. In this article, the proposed metrics are used for evaluation of communities of co-authors that were extracted from DBLP dataset (see Section 5).

The study of the dynamic evolution is relatively new subject in the research of the social communities. The research of this paper is focused to study the communities extracted from the DBLP dataset and their dynamic grow in time. The short introduction to the social network is described in the Section 2 and general concept of the DBLP is shown in the Section 3. The Section 4 contains description of dynamic metrics and in the Section 5 is shown practical example of using these metrics on communities of co-authors from DBLP. Also in Section 5 is described algorithm for Extraction of Communities of Co-authors in time.

2 Social Networks

A social network (SN) is a set of people or groups of people with similar pattern of contacts or interactions such as friendship, co-working, or information exchange [10]. The World Wide Web, citation networks, human activity on the internet (email exchange, consumer behaviour in e-commerce), physical and biochemical networks are some examples of social networks. Social networks are usually represented by graphs, where nodes represent individuals or groups and lines represent relations among them. Mathematicians and some computer scientists usually describe these networks by means of graph theory [7].

Social network analysis (SNA) is a collection of methods, techniques and tools that aim to analyse the social structures and relational aspects of these structures in a social network [11]. The study of social networks is a quite old discipline. Many studies oriented to the analysis of social networks have been provided. The datasets used in these studies are obtained by using questionnaires. In contrast to previous SNA research, contemporary provided, and more structured approaches, are based on the automated way of research. In the late 1990s, development of new information and communication technologies (such as internet, cellular phones) enabled the researchers to construct large-scale networks using the data collections stored in e-mail logs, phone records, information system logs or web search engines.

Community detection is an important aspect in discovering the complex structure of social networks. A community is defined as a subset of nodes within the network such that connections between the nodes are denser than connections with the rest of

the network [10]. Community structure can be defined using modules (classes, groups or clusters etc.).

3 Digital Bibliography Library Project

DBLP (Digital Bibliography Library Project) is a computer science bibliography database hosted at University of Trier, in Germany. It was started at the end of 1993 and listed more than 2.1 million publications in January 2013. These articles were published in Journals such as VLDB, the IEEE and the ACM Transactions and Conference proceedings [4]. DBLP has been a credible resource for finding publications, its dataset has been widely investigated in a number of studies related to data mining and social networks to solve different tasks such as recommender systems, experts finding, name ambiguity, etc. Even though, DBLP dataset provides abundant information about author relationships, conferences, and scientific communities, it has a major limitation that is its records provide only the paper title without the abstract and index terms.

Many experts focuses on the task of finding persons with high level of experience on a specific topic. To achieve this objective researchers approached this task mainly in three different ways. The first group applied an information retrieval techniques to solve it [1], the authors of this paper proposed a weighted language model, which introduces a document prior probability to measure the importance of the document written by an expert. The second group approached this task using social network analysis metrics [12], in this study a large online help seeking community, the Java Forum, was analysed using social network analysis methods and a set of network-based algorithms, including PageRank and HITS. While the third group used a hybrid approach of information retrieval and social network analysis for finding academic experts [13]. In [13] the authors created a local information document for each person to measure his initial level of experience on a topic using information retrieval models. Then they applied propagation on the graph of experts to update his level of expertise according to his relations with the other nodes. In the article [2], the authors focused on the detection of communities with the use of spectral clustering. This algorithm was used in the article [6] to find the communities in a subnetworks that were defined by the selected terms (from the whole DBLP).

4 Dynamic network analysis

Dynamic network analysis (DNA) varies from traditional social network analysis. DNA could be used for analysis of the non static information of nodes and edges of social network. DNA is a theory in which relations and strength of relations are dynamic in time and the change in the one part of the system is propagated through the whole system, and so on. DNA opens many possibilities to analyse and study the different parts of the social networks. We can study behaviour of individual communities, persons or the whole graph of the social network. The paper is focused to analyse the behaviour of communities extracted from DBLP and divided by time. The proposed approach which use dynamic metrics is inspired by work of Palla et al. [8].

The auto-correlation function $C(t)$ is used to quantify the relative overlap between two states of the same community $A(t)$ at t time steps apart:

$$C(t) = \frac{|A(t_0) \cap A(t_0+t)|}{|A(t_0) \cup A(t_0+t)|}, \tag{1}$$

where $|A(t_0) \cap A(t_0+t)|$ is the number of common nodes (members) in $A(t_0)$ and $A(t_0+t)$, and $|A(t_0) \cup A(t_0+t)|$ is the number of nodes in the union of $A(t_0)$ and $A(t_0+t)$.

The stationarity of community is defined as the average correlation between subsequent states:

$$\zeta = \frac{\sum_{t=t_0}^{t_{max}-1} C(t,t+1)}{t_{max}-t_0}, \tag{2}$$

where t_0 denotes the birth of the community, and t_{max} is the last step before the extinction of the community. Thus, $(1-\zeta)$ represents the average ratio of members changed in one step.

Authors of the paper [8] found that the auto-correlation function decays faster for the larger communities, showing that the membership of the larger communities is changing at a higher rate. In contrast, they said that small communities change at a smaller rate with their composition being more or less static. The stationarity was used to quantify static aspect of community evolution.

5 Evolution of Co-authors Communities

To create our experiments and to count dynamic metrics we generate DBLP subgraphs of selected terms for each year in which this term occurs. Generating of these subgraphs of DBLP authors is described in the following section. This final set of subgraphs is input for our experiments and to count dynamic metrics.

5.1 Extraction of Communities of Co-authors in Time

For the experiments we used a data collection of publications and their authors from the DBLP server [1]. When processing XML dataset we analysed records for the following publication types: *article*, *inproceedings* and *incollection*. During the experiment 2,055,469 articles (set *Articles*), 1,182,363 authors (set *Authors*) were indexed and 308,933 terms from titles of articles (set *Terms*) were extracted. A set of *Terms* contains both terms lemmatized by Porter's algorithm [9] and their forms without lemmatization. For each article we store informations about authors, key for DBLP collection, date when it was added to the DBLP collection and publication year. For an author we register his ID, simplified name for information retrieval, special form of his name for the DBLP collection, number of articles and links to the most important terms of the author. Furthermore, we use a matrix of articles and their terms $M^{Articles \times Terms}$.

[1] DBLP dataset: http://dblp.uni-trier.de/xml/ - downloaded October 2012

```
Example of Article
*******
Key: reference/social/SlaninovaMDOS10
Date: {1/1/2010 12:00:00 AM}
Id: 876067
MDate: {11/13/2011 12:00:00 AM}
Authors Count: 5
```

Before creating a subgraph, we need to determine the set of terms, which we will be searching for. These terms represent articles we are interested in. We will denote this set as *Query*. It can contain both terms with or without the lemmatization. We use both forms because anyone can come across the need to look up words in their original form. As an example, the word *modularity* in social networks means something different than the base form *modul* obtained by the lemmatization. After we identified the terms, we need to get the articles defined by these terms. These articles $Articles_Q$ are determined by the non-zero values in the matrix M in those columns, that match the searched terms (OR query). If we want to select only those articles in whose titles contains all entered terms (AND query), then we must remove such articles from the set $Articles_Q$ which have some of the term missing in the title.

The set of the years in which the articles were published in the set $Articles_Q$ we denote as Y. From the set of articles $Articles_Q$ we select set of authors $Authors_y$ who published together, for each year $y \in Y$. Now for every year $y \in Y$ we create graph $G_y(Authors_y, E)$, where E represents strength of authorship.

Dynamic metrics described in the Section 4 are generally metrics used to evaluate the characteristics of the community. About such community, we have to know that it changes over time and also we should have information on how the community looked at each time step of its existence. Therefore to get the information about the communities and their changes in time from subgraph of the authors, we need to execute a series of steps which are described below.

Algorithm for Finding Component Evolution in Time

(I) *Creating the longest continuous consecutive time chain of graphs G_y*
Input graphs may have different time intervals between them. But for the next step we need to choose the longest consecutive time period with one year interval.
For example:
Input graphs: G_{1998}, G_{1999}, G_{2002}, G_{2003},G_{2004}, G_{2005},G_{2006},G_{2007}, G_{2012}.
For processing we use this set of graphs: G_{2002}, G_{2003},G_{2004}, G_{2005},G_{2006},G_{2007}.

(II) *Finding connected components of the subgraph*
Graphs from the previous step are non connected. We search for all the connected components to get components for each year with which we will continue to work.

(III) *Create chain of the connected components across all time steps*
1. We choose the first largest component c from the graph in the first time step.
2. According to the following rules we select next component (follower) in the next time step based on the current component c. We denote this component as similar component. We are looking for the components which has the biggest number of the same nodes as the current component c and for selection we have to choose one of the following options:

(a) If only one similar component is found we denote it as follower.
(b) If more than one similar components are found we denote the biggest one as follower.
(c) If no component is found we choose as a follower the biggest existing component in this time step.

3. Step 2 is repeated for each time step except the last one.

Basically we are talking about the components that consist of the DBLP authors and links between them which are formed on the basis of the common interest - the same terms in the titles of their articles. Therefore we can say that our components are the communities of the co-authors. Due to the above described algorithm, we prepare the set of consecutive components. We assume that this set represents the development of one community over time.

This idea allows us to calculate dynamic metrics described in the Section 4. Recall that the auto-correlation is calculated for each of the two states of the same community, followed with computed value of stationarity.

5.2 Experiments

To demonstrate experiments, we choose terms: ”elearning”, ”elearning teach blackboard”, ”elearning teach moodle”, ”mysql”, ”oracle”, ”social network”, ”dynamic social network”, ”social network analysis”. Basic properties of the communities found for each set are described in the table, where we present the count of time steps for each community.

Terms	Count of time steps	Year from	Year to
elearning	12	2001	2012
elearning teach blackboard	43	1971	2013
elearning teach moodle	43	1971	2013
mysql	5	2008	2012
oracle	33	1981	2013
social network	17	1997	2013
dynamic social network	10	2003	2013
social network analysis	17	1997	2012

Table 1. Communities of co-authors developed in time

Evolution of communities of co-authors in the time are demonstrated in the Figures 1 and 2. These figures show changes of counts of members of each community in time.

In Figure 1 on the left, we can see a development of the three communities, which published in similar areas, namely ”social network”, ”dynamic social network” and ”social network analysis”. If we look at the change of the curves of authors in communities that deal with ”social network” and ”social network analysis”, we will notice that curves from 1997 to 2009 look similar. In 2009, we can notice a great interest in the generic term ”social network”. According to information shared by Facebook provider

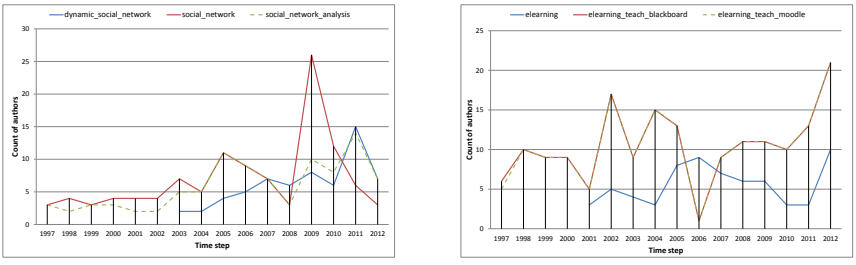


Fig. 1. Evolution of communities of co-authors for the terms *"social network", "dynamic social network", "social network analysis" and "elearning", "elearning teach blackboard", "elearning teach moodle"*

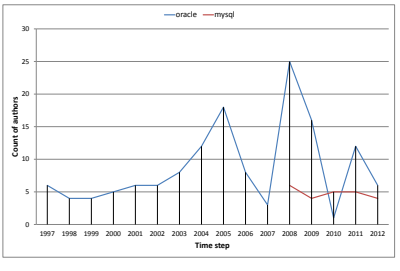


Fig. 2. Evolution of communities of co-authors for the terms *"mysql", "oracle"*

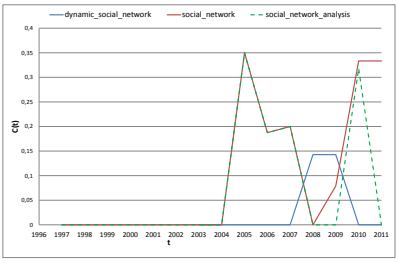
in 2009 [2], there was the largest detected increase of new users on Facebook. In 2009, around 150 million new users have joined the social networking site Facebook. In the following years, the number of newly connected users varied from 5 to 50 millions per year.

Since 2009, interest in generic term "social network" began to decline strongly. On the other hand, interest in terms "dynamic social network" and "social network analysis" had increased. At the same time, these two curves began to grow similarly.

We would like to draw attention to an important property of value of auto-correlation. Auto-correlation is always computed for the community in a time interval t to the change of the community in the following time slot $(t+1)$. Because of this property, we show the results until 2011 in Figure 3 since the value of auto-correlation for 2012 can be calculated correctly only at the end of 2013.

On the left side of Figure 3, we present auto-correlation values for communities "social network", "dynamic social network" and "social network analysis". The higher the

[2]Number of active users at Facebook over the years, http://news.yahoo.com/number-active-users-facebook-over-230449748.html



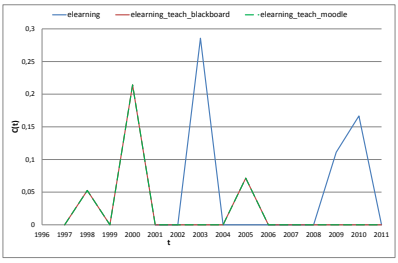


Fig. 3. Auto-correlations of communities of co-authors for terms *"social network", "dynamic social network", "social network analysis" and "elearning", "elearning teach blackboard", "elearning teach moodle"*

auto-correlation is, the more authors in the community in these time periods had stable interest in publishing together with someone else. This means in our case publishing together in the same area of interest that was initialized by the terms. According to the auto-correlation curves, there was stable interest in "social network" in 2004 which then continuously decreased until 2009. From 2009 onwards we can see a stable growth of interest in publishing in "social network". From 2007 to 2010, there is evident growth of interest in the field of "dynamic social network". However, it is smaller than that of the generic term "social network". For the community "social network analysis", we can follow a similar stability evolution of the authors who published in the area of "social networks".

We can create the same analysis for the auto-correlation curves of communities formed by terms "elearning", "elearning teach blackboard" and "elearning teach moodle", shown on the right side of the Figure 3 . In this analysis can be noticed an interesting factor that from 2011 to 2012, the community which deals with "elearning" has the largest value of auto-correlation. We could say that it is experiencing a period of steady state of authors who publishes in this area.

In the Figure 4, we show the values of stationarity for all communities, which we analysed in our experiments. In general, this value characterizes the degree of variability of community in time. The larger the value of stationarity is, the more the community is stable and static. On the other hand, the smaller value indicates a community more dynamic and more changeable in time. In the Figure 4, we see that the largest value of stationarity has the community publishing about "social network", "oracle", "social network analysis", "elearning". But if we look at the data, we may notice that the communities dealing with "social network" a "social network analysis" are relatively young, and therefore their values of stationarity are higher than in the older communities. Communities dealing with "dynamic social network", "elearning teach blackboard" a "elearning teach moodle" are more dynamic in the sense that only a few authors have published in this area for a time.

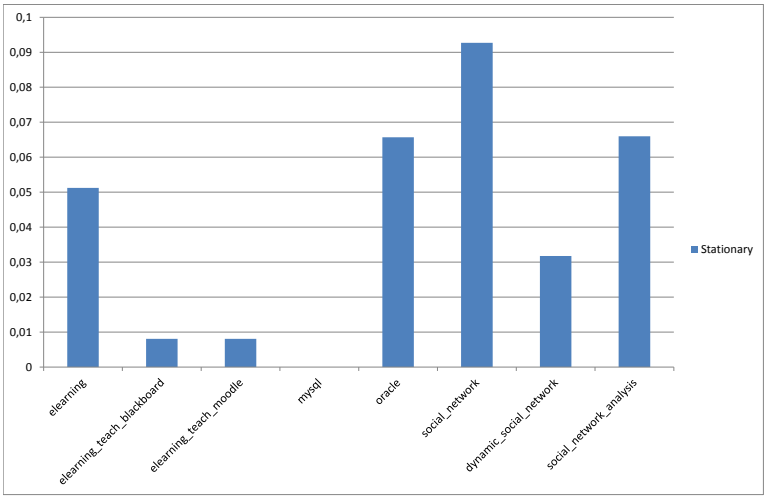


Fig. 4. Stationarity of communities of co-authors for terms *"elearning", "elearning", "elearning teach blackboard", "elearning teach moodle", and "social network", "dynamic social network", "social network analysis"*

6 Conclusion

The research presented in this paper is oriented to analysis of communities of co-authors evolution formed by terms on DBLP. In the paper, the analysis of evolution of co-authors in the communities was presented, with the focus to their growth. The method for evaluation of the stability of authors' interests in the communities extracted from DBLP was described. Moreover, the method for identification of dynamic or static communities in the time was presented. Experiments have been demonstrated on the network of co-authors. Naturally, presented methods can be used for other different networks and another types of communities.

The step Number III is one of the most important steps in the algorithm presented in the Section 5.1, because it defines which components represent an image of one component in different time periods. In future, we want to enrich our experiments by changing this step of the presented algorithm. Together with condition for a particular user incorporated into this step, it gives a completely different view on the issue of selecting the components. Analysis of the evolution of community formed around a user brings the opportunity to research and analyse not only dynamic properties of the community itself but also the possibility of studying the characteristics of the users or the analysis of evolution in individual cases.

Acknowledgment

This work was supported by SGS, VSB – Technical University of Ostrava, Czech Republic, under the grant No. SP2013/167 Analysis of Users' Behaviour in Complex Networks.

References

1. H. Deng, I. King, and M. R. Lyu. Formal models for expert finding on dblp bibliography data. *2008 Eighth IEEE International Conference on Data Mining*, pages 163–172, 2008.
2. P. Drazdilova, J. Martinovic, and K. Slaninova. Spectral clustering: Left-right-oscillate algorithm for detecting communities. In M. Pechenizkiy and M. Wojciechowski, editors, *New Trends in Databases and Information Systems*, volume 185 of *Advances in Intelligent Systems and Computing*, pages 285–294. Springer Berlin Heidelberg, 2013. 10.1007/978-3-642-32518-2_27.
3. J. K. Lars Backstrom, Dan Huttenlocher. Group formation in large social networks: membership, growth, and evolution. *Science*, pages(9):44–54, 2006.
4. M. Ley. The dblp computer science bibliography: Evolution, research issues, perspectives. *LNCS*, 2476:1–10, 2002.
5. F. S. Mansoureh Takaffoli, Justin Fagnan and O. Zaiane. Tracking changes in dynamic information networks. *2011 International Conference on Computational Aspects of Social Networks CASoN*, pages 94–101, 2011.
6. S. Minks, J. Martinovic, P. Drazdilova, and K. Slaninova. Author cooperation based on terms of article titles from dblp. In *IHCI2011*, 2011.
7. M. E. J. Newman. The structure and function of complex networks. *SIAM Review*, 45(2):58, 2003.
8. G. Palla, A. lászló Barabási, T. Vicsek, and B. Hungary. Quantifying social group evolution. *Nature*, 446:664–667, 2007.
9. M. F. Porter. Readings in information retrieval. chapter An algorithm for suffix stripping, pages 313–316. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1997.
10. F. Radicchi, C. Castellano, F. Cecconi, V. Loreto, and D. Parisi. Defining and identifying communities in networks, Feb 2004.
11. J. Scott. *Social Network Analysis*. Newbury Park CA: Sage, 1992.
12. J. Zhang, M. S. Ackerman, and L. Adamic. Expertise networks in online communities: Structure and algorithms. In *WWW '07: Proceedings of the 16th International Conference on World Wide Web*, pages 221–230, New York, NY, USA, 2007. ACM.
13. J. Zhang, J. Tang, and J. Li. Expert finding in a social network. *Advances in Databases Concepts Systems and Applications*, 4443:1066–1069, 2007.
14. Y. Q. Zhixing Huang, Yan Yan and S. Qiao. Exploring emergent semantic communities from dblp bibliography database. *2009 International Conference on Advances in Social Network Analysis and Mining*, pages 219–224, 2009.

A Linguistic Method into Stemming of Arabic for Data Compression

Hussein Soori, Jan Platoš, and Václav Snášel

Faculty of Electrical Engineering and Computer Science
VSB-Technical University of Ostrava, Czech Republic
{sen.soori, jan.platos,vaclav.snasel}@vsb.cz

Abstract. Creating good stemming rules for the Arabic language comes from the importance of Arabic language as the sixth most used language in the word. Stemming is very important in information retrieval, data mining and language processing. With Arabic having complex morphology and grammatical properties, this poses a challenge for researchers in this field. In this paper, we try to use an online morphological parser to distinguish parts of speech (POS), and then set some extracting rules to produce stems, and finally, mismatch these stems with an electronic dictionary. As a pilot study for this method, in this paper we deal with three POS: nouns, verbs and adjectives.

Keywords: Stanford Online Parser, data compression for Arabic, Arabic natural language processing, Arabic data mining, Arabic morphology, stemming of Arabic.

1. Introduction

The rapidly growing number of computer and Internet users in the Arab world and the fact that the Arabic language is the sixth most used language in the world today creates a demand for more research in the area of data mining and natural language processing in Arabic language. Another two factors maybe that Arabic alphabet is the second-most widely used alphabet around the world - Arabic script has been used and adapted to such diverse languages as Amazigh (Berber), Hausa, and Mandinka (in West Africa), Hebrew, Malay (Jawi in Malaysi and Indonesia), Persian, the Slavic tongues (also known as Slavic languages), Spanish, Sudanese, and some other languages, Swahili (in East Africa), Turkish, Urdu [10], and that Arabic is one of the six languages used in the United Nations [11] after the Latin alphabet.

1.1 Arabic Complex Morphological and Grammatical Properties

A few challenges may face researchers as for as the special nature of Arabic script is concerned. Arabic is considered as one of the highly inflectional languages with complex morphology. Unlike most other languages, it is written horizontally from right to left. It consists of 28 main letters. The shape of each letter depends on its position in a

word—initial, medial, and final. There is a fourth form of the letter when written alone. One example of this can be given for the letter (ع) as follow:

Initial	Medial	Final	Separate
عـ	ـعـ	ـع	ع

Fig. 1. Arabic Alphabets

Moreover, the letters alif, waw, and ya (standing for glottal stop, w, and y, respectively) are used to represent the long vowels a, u, and i. This is very much different from Roman alphabet which is naturally not linked. Other orthographic challenges can be the the persistent and widespread variation in the spelling of letters such as hamza (ء) and ta' marbuTa (ة), as well as, the increasing lack of differentiation between word-final ya (ي) and alif maqSura (ى). Typists often neglect to insert a space after words that end with a non-connector letter such as ر , ز , و [3]. In addition to that, Arabic has eight short vowels and diacritics (َ , ّ , ٌ , ٍ , ً , ْ , ُ , ِ). Typists normally ignore putting them in a text, but in case of texts where typists do put them, they are pre-normalized –in value- to avoid any mismatching with the dictionary or corpus in light stemming. As a result, the letters in the decompressed text, appear without these special diacritics.

Diacritization has always been a problem for researches. According to Habash [12], since diacritical problems in Arabic occur so infrequently, they are removed from the text by most researchers. Other text recognition studies in Arabic include, Andrew Gillies *et al.* [11], John Trenkle *et al*. [30] and Maamouri *et al.* [20].

Other than letters, another factor determain the word identity and in many instances can change the meaning and part of speech. This factor is the eight short vowels and diacritics (َ , ِ , ُ , ْ , ً , ٍ , ٌ , ّ). An example for (رجل) is given in the following table where we can see the total change in word category and meaning as a result of adding the diactricals which resulted in producing three different words in meaning and three different parts of speech for the same three letter رجل :

Word	Meaning	Part of Speech
رجُلُ	man	noun (subject)
رجُلَ	man	noun (object)
رِجْل	foot	noun
رَجِلَ	to go on foot (rather than, e. g., ride a bike)	verb

Never the less, it is always advised that these vowels and diacritics are often normalized before processing in most light stemming or morphological approaches [4]. Mainly the reasons for not including them in the word processing is the claim that they do occur so infrequently, and that in Modern Standard Arabic (MSA), people

tend not to use them and, as a result of that, the meaning is left for the native speaker's intuition, or , in some cases, can be determined from the context. This problem is still waiting for a challenging attempt where the processor is ready to process words with or without diacritics, without needing to normalize words.
Another morphological feature in Arabic is that, unlike Roman letters which are separated naturally, Arabic has an agglutinated nature(as mentioned above) where letters are linked to each other in some cases, while unlinked in some other case, depending on position of the letter in the root, stem and word level. For example, in English the pronoun (he) in (he plays) is separated from the following noun (plays), while in Arabic the pronoun is represented by the letter (ي) which is linked to the root verb لعب to form يلعب (he plays). The same is true when it comes to different kinds of Affixes.
Arabic has four types of affixes. Prefixes: these are letters (normally one) that change the tense of the verb from past to present, such as the letter (ي) in case of the verb لعب and يلعب above. Suffixes: these represent the inflectional terminations (endings) of verbs, as well as, the female and dual/plural markers for the nouns. Postfixes: these are the pronouns attached at the end of the word. Antefixes: these are prepositions agglutinated to the beginning of words.

1.2 The Problem at Hand:

This paper is trying to improve the rules for stemming of Arabic texts for data compression. A few different linguistic methods were used by us in the past, for example: the vowel letter method [2]. This method was mainly dependent on syllabification of words and focused on splitting words according to vowel letters. The second approach [8] was a simple approach into stemming rules, where 4 category of words were selected (nouns, verbs, adjectives and adverbs) from short news item texts. These two approaches produced some good results. However, two major problems showed up.
The first problem had to do with parts of speech (POS) recognition problem. For example, the verb يلعب (plays) starts with the letter (ي). In Arabic, adding the suffix (ي) is a very common way to change the word from its past form into its present form. When some rules are set to remove the letter (ي) so to produce the root form of لعب , these rules always removed the letter (ي) from other POS as well, such as the word يمن (Yemen) where the letter (ي) is part of the root word .
The second problem occurs within the sub-POSs when, for example, trying to remove the determiner ال (the definite article 'the') from common nouns as in الطالب (the student). The rules set remove the ال from all nouns including proper nouns such as, المانيا (Germany) where the ال is part of the original noun and not a determiner.
For these reasons, in this paper we try to use Stanford online [9] to better categorize the different POS and later to be mismatch the output words -after stemming- with an elctronic dictionary.

1.3 The Stanford Online Parser

The Stanford parser is a powerful online parser that parses texts in three languages: Arabic, Chinese and English. This parser is using dependency grammar. The Arabic parts of the parser [9]is depending on the Penn Treebank project that was launches in

2001 in the University of Pennsylvania and headed by Prof. Mohamed Maamouri. According to this corpus documentation [10], this corpus is designed for those who study or use languages professionally or academically, as well as, for those who need text corpora in their work. The Penn Arabic Treebank is particularly suitable for language developers, computational linguists and computer scientists who are interested in various aspects of natural language processing.

Table 1: English transliteration of Arabic alphabets

Arabic Alphabet	Transliteration	Arabic Alphabet	Transliteration
ا	alif	ع	Ayn
ب	baa	غ	ghayn
ت	ta	ف	faa
ث	tha	ق	qaaf
ج	jiim	ك	kaaf
ح	haa	ل	laam
خ	kha	م	miim
د	daal	ن	nuun
ذ	thal	هـ	haa
ر	raa	ة	taMarboota
ز	zay	و	waaw
س	siin	لا	laamAlif
ش	shiin	ء	hamza
ص	Saad	ئ	hamzaONyaa
ض	Daad	ؤ	hamzaONwaaw
ط	Taa	ي	yaa
ظ	Dhaa	ى	alifMaqsoora

1.4 The Arabic Alphabets Transliteration System

In this study, we use a transliteration system for Arabic Alphabets so to enable non-Arabic speakers identify Arabic alphabets and to to understand the rules proposed. A legend of Arabic Alphabets and their English transliterations is provided in Table 1.

2. Stemming Rules

According to Stanford Online Parser for Arabic language, there are 27 different POSs. In this paper, a number of rules are set for 3 main POSs: nouns, verbs and adjectives as follows:
The rule for every POS or sub-POS is divided into steps as shown below. Every step is to be implemented in the order of numbering:

Specifications
W – any word or its part (word referes to any POS in the rule: noun, verb, adjective, etc.)
[] – arabic letter
Ins(x, y) – return true when x is anywhere in y
|x| - length of word x
[x]W – letter x is at the beginning of the word

Nouns Rules:
a) DTNN: determiner + singular common noun

Step 1: [alif laamAlif laamAlif]W -> [alif laam]W
Step 2: [alif laamAlif]Wxy -> [alif laam]Wy

b) DTNNP: determiner + singular proper noun

Step 1: [alif laam]W -> W

c) DTNNS: determiner + plural common noun

Step 1: [alif laam]W -> W

d) NNPS: common noun, plural or dual

Step 1: W[ta] -> W
W[yaa nuun] -> W
Step 2: |W| < 5 -> W[taMarboota]
Step 3: W[waaw][taMarboota] -> W[taMarboota]

Verbs Rules:

a) VBD: perfect verb (*nb: perfect rather than past tense)**

Step 1: |[waaw]W|>2 -> W
Step 2: W[alif] -> W
W[ta] -> W
W[waaw nuun] -> W
Step 3: W[alif haa] -> W[alifMaqsoora]
W[ta haa] -> W[alifMaqsoora]

b) VBN: passive verb (*nb: passive rather than past participle)**

Step 1: [yaa]W -> W
Step 2: |W| = 4 & [ta]W -> [alif]W

c) VBP: imperfect verb (*nb: imperfect rather than present tense)**

Step 1: [ta]W -> W
[ta ta]W -> W
[yaa]W -> W
Step 2: W[waaw] -> W
Step 3: [nuun]W -> W
[waaw nuun]W -> W
[haa]W -> W
[haa alif]W -> W
Step 4: |W| = 2 -> W[alifMaqsoora]
Step 5: W[yaa] -> [alif]W[alifMaqsoora]
Step 6: [siin]W & ins(W, [ta]) -> [alif][siin]W
Step 7: W[waaw laam] -> W[alif laam]
W[waaw laam waaw nuun] -> W[alif laam]
W[waaw nuun] -> W[alif laam]
Step 8: [nuun][ta]W & |[nuun][ta]W| > 3 -> [nuun]W

Adjectives Rules:

a) DTJJ: determiner + adjective

Step 1: [alif laam]W -> W
Step 2: W[taMarboota] -> W

3. Experiments

The suggested rules must be tested against real data. For this purpose, we use some news articles, from the BBC Arabic and Al Jazeera Arabic news portals. These articles are parsed by Stanford Online Parser and the results are shown in table 2. In the

following table, repeated words are deleted and sample words of every POS or sub-POS are shown in the table.

Table 2. List of words used in our experiments

Nouns
a) DTNN: determiner + singular common noun
البحر البنزين البيئة الجرف الحد الدفاع السيناتورين الشرق الشؤون البر المصادر السابق الحكوم المحيط النفط الوصول العالم الاحتياطي التنقيب العام الدولة الابار الاحتياطي الامر الامور الاميال الانسان بالكلاب الطاقة الطلب العواصف المثبت المحيط المشاغل الموارد المياه النفط الوحل الانواع البيئة العاصمة الاخطار الفصائل البيئة الدردشة السهل
b) DTNNP: determiner + singular proper noun
الانترنت البرازيل الدوحة العاج المكسيك
c) DTNNS: determiner + plural common noun
الامريكيين الاولويات الجمهوريون الديموقراطيون الشركات العشرات لتحقيق للحماية المحافظون المشترين المعاملات الملوثات المنتجات المندوبون المنصات الولايات
d) NNPS: common noun, plural or dual
تغييرات تقديرات جماعات سنوات طبقات طنين عشرات عمليات كميات مجتمعات مقترحات منصات
Verbs
a) VBD: perfect verb (*nb: perfect rather than past tense)**
اجراها اجرته ادى استخرجت اصبح اصبحت اقترع ايد بامكاننا تسبب تسربا جعلت فقد كان مضى وتزيد وقال وقد وقدمت ويضيفون ويقول
b) VBN: passive verb (*nb: passive rather than past participle)**
يوجد تسحب يرجح تستخدم يدرج
c) VBP: imperfect verb (*nb: imperfect rather than present tense)**
تبدو تتنافس تتوزع تتوقف تتيحها تحتوي تربض تستخرج تسجل تشارك تشير تصبح تطا تطفو تعد تلاحظه تنتج تهدد يبدو يبلغ يتعرض يتفوق يتم يتناولها يحاولون يحتوي يختبئى يخرجها يزال يساهم يستحق يستفيد يسمون يشكل يصبح يقترع يقع يقول يقولون يكاد يكون يمارس يمثل يمكن ينتشر ينتهي ينجح يهدد
Adjectives
a) DTJJ: determiner + adjective

الاستقلالية الاشعاعية الامريكي الامريكية الاولى البرية البيئية التجارية الجاري الجديد الحمراء الحيوانية الخارجي الداخلية الدولي الدولية الطبيعية العالمي العالمية القاري القانونية القطبية القطرية المتبقية المتحدة المحلية المحمية المرجانية المهددة المهيمن النادرة النفطية الواسعة

Before any rule is applied, all words must be normalized and preprocessed. We store all words in plain text files using codepage 1256 – Arabic. Because all our software is written in C+, we read these text files into Unicode representation.

Our results for the nouns list are depicted in Tables 3, 4, 5 and 6. The results for the noun rules produced very good results in case of DTNNP and DTNN. Very few undesirable results were produced because some words were wrongly parsed by the parser such as (بالكلاب). As for DTNNS, some more rules needed to deal with the plural and dual suffixes. NNPS produced very good results.

Table 3. Processed Nouns - DTNN: determiner + singular common noun

حكوم سابق مصادر بر شؤون شرق سيناتور دفاع حد جرف بيئة بنزين بحر طاقة بالكلاب مر ميل البر دولة عام تنقيب عالم وصول نفط محيط عاصمة بيئة النوع وحل نفط مياه موارد مشاغل محيط مثبت عواصف طلب دردشة سهل بيئة فصائل الخطر

Table 4. Processed nouns - DTNNP: determiner + singular proper noun

عاج مكسيك دوحة برازيل انترنت

Table 5. Processed nouns - DTNNS determiner + plural common noun

لتحقيق عشرات شركات ديموقراطيون جمهوريون اولويات امريكيين منصات ولايات مندوبون منتجات ملوثات معاملات مشترين محافظون للحماية

Table 5. Processed Nouns -NNPS: common noun, plural or dual

مقترح منصة مجتمع كمية عملية عشرة طنة طبقة سنة جماعة تقدير تغيير

The verbs' rules results are depicted in Tables 7, 8 and 9. The verbs' rules produced good results in case of VBD and VBN. However, in case of VNP, a few bad results show up and the rules have to be enhanced in the future.

Table 7. Processed Verb - VBD: perfect verb

جعلت تسرب تسبب بامكانن اید اقترع اصبح اصبح استخرج ادى اجرى اجرى يضيف يقول قدمت قد قال تزيد مضي كان فقد

Table 8. Processed verbs – VBN: passive verb

درج استخدم رجح سحب وجد

Table 9. Processed Verb – VNP: imperfect verb

طای اصبح شارك سجل استخرج ربض احتوى تيحها توقف توزع تنافس بدى ختبئى احتوى حاال تناولها تمى تفوق تعرض بلغ بدى تجى لاحظ عدى طفى كال كاد قال قال قعى اقترع اصبح شكل استفيد استحق ساهم زال خرجها مكن مثل مارس

The results for the adjectives' rules are depicted In Table 10. Almost all rules made for adjectives produced successful results.

Table 10. Processed adjectives - DTJJ

جديد جاري تجاري بيئي بري اولى امريكي امريكي اشعاعي استقلالي قاري عالمي عالمي طبيعي دولي دولي داخلي خارجي حيواني حمراء نادر مهيمن مهدد مرجاني محمي محلي متحد متبقي قطري قطبي قانوني نفطي واسع

4. Conclusion

In this paper we set rules for POS and to parse our training data, we used Stanford Online Parser for Arabic language, which identifies 27 different POSs. In this paper, the rules set are for 3 main POSs: nouns, verbs and adjectives. Every rule for every POS or sub-POS is divided into one or more steps.

The results for the noun rules produced very good resuts in case of DTNNP and DTNN. Very few undesirable results occur because some words were wrongly parsed by the parser such as (بالكلاب). As for DTNNS, some more rules needed to deal with the plural and dual suffixes. NNPS produced very good results. The verbs' rules results are depicted in Tables 7, 8 and 9. The verbs' rules produced very good results in case of VBD and VBN. However, in case of VNP, a few bad results show up and the rules have to be enhanced in the future. The results for the adjectives's rules are depicted In Table 10. Almost all rules made for adjectives produced very good results. Most errors occurred in case of VBP. However, the overall evaluation of these rules proved that the rules produced very good results. In the future, these rules must be improved and enhanced to include more POSs and should be tested against wider variety of vocabulary and bigger corpora.

Acknowledgments: This work was partially supported by the Grant Agency of the Czech Republic under grant no. P202/11/P142, SGS in VSB – Technical University of Ostrava, Czech Republic, under the grant No. SP2013/70, and has been elaborated in the framework of the IT4Innovations Centre of Excellence project, reg. no. CZ.1.05/1.1.00/02.0070 supported by Operational Programme 'Research and Development for Innovations' funded by Structural Funds of the European Union and state

budget of the Czech Republic and by the Bio-Inspired Methods: research, development and knowledge transfer project, reg. no. CZ.1.07/2.3.00/20.0073 funded by Operational Programme Education for Competitiveness, co-financed by ESF and state budget of the Czech Republic.

References

1. Encyclopedia Britannica Online. Alphabet. Online (2011). URL: http://www.britannica.com/EBchecked/topic/17212/alphabet
2. H. Soori, J. Platos, V. Snasel, H. Abdulla, in Digital Information Processing and Communications, Communications in Computer and Information Science, vol. 188, ed. By V. Snasel, J. Platos, E. El-Qawasmeh (Springer Berlin Heidelberg, 2011), pp. 97{105. URL http://dx.doi.org/10.1007/978-3-642-22389-1 9. 10.1007/978-3-642-22389-1 9
3. T. Buckwalter, in Arabic Computational Morphology, Text, Speech and Language Technology, vol. 38, ed. by N. Ide, J. Veronis, A. Soudi, A.v.d. Bosch, G. Neumann (Springer Netherlands, 2007), pp. 23{41. URL http://dx.doi.org/10.1007/978-1-4020-6046-5 3.10.1007/978-1-4020-6046-5 3
4. N.Y. Habash, Synthesis Lectures on Human Language Technologies 3(1), 1 (2010). DOI 10.2200/S00277ED1V01Y201008HLT010.
   URL http://www.morganclaypool.com/doi/abs/10.2200/S00277ED1V01Y201008HLT010 (last accessed 10/12/2012)
5. A. Gillies, E. Erl, J. Trenkle, S. Schlosser, in Proceedings of the Symposium on Document Image Understanding Technology (1999)
6. J. Trenkle, A. Gilles, E. Eriandson, S. Schlosser, S. Cavin, in Symposium on Document Image Understanding Technology (2001), pp. 159{168
7. M. Maamouri, A. Bies, S. Kulick, in Proceedings of the British Computer Society Arabic NLP/MT Conference (2006).
8. Soori, H. , Platoš, J. , Snášel, V.: Simple stemming rules for Arabic language, Advances in Intelligent Systems and Computing, Volume 179 AISC, 2012, Pages 99-108, ISBN: 978-364231602-9
9. Spence Green and Christopher D. Manning. 2010. Better Arabic Parsing: Baselines, valuations, and analysis. In 23rd Conference on Computational Linguistics, pages 394–402, Beijing, China.
10. http://www.ircs.upenn.edu/arabic/Jan03release/README.txt (last accessed 10/03/2013)
11. http://www.un.org/ (last accessed 10/03/2013)

Searching Time Series Based On Pattern Extraction Using Dynamic Time Warping

Tomáš Kocyan[1], Jan Martinovič[1], Pavla Dráždilová[2], and Kateřina Slaninová[2]

[1] VŠB - Technical University of Ostrava,
IT4Innovations,
17. listopadu 15/2172, 708 33 Ostrava, Czech Republic
{tomas.kocyan,jan.martinovic}@vsb.cz
[2] VŠB - Technical University of Ostrava,
Department of Computer Science,
17. listopadu 15/2172, 708 33 Ostrava, Czech Republic
{pavla.drazdilova,katerina.slaninova}@vsb.cz

Abstract. Many types of data collections processed by time series analysis often contain repeating similar episodes (patterns). If these patterns are recognized, then they may be used for instance in data compression, for prediction or for indexing large collections. Extraction of these patterns from data collections with components generated in equidistant time and in finite number of levels is now a trivial task. The problem arises for data collections that are a subject to different types of distortions in all axes. In this type of collections, the found similar episodes do not have to be exactly the same; they can differ in time, shape or amplitude. In these cases, it is necessary to pick the suitable one from each group of similar episodes and to declare it as a representative member of the whole group. This paper discusses the possibilities of using the Dynamic Time Warping (DTW) method for deriving the representative member of a group of similar episodes that are subjects to the previously mentioned distortions. The paper is also focused on providing a suitable mechanism for more effective searching of distorted time series.

Keywords: Dynamic Time Warping, Time Series, Pattern Mining

1 Introduction

Time series analysis covers methods for analysis of time series data with a focus on extraction of various types of information like statistics and other characteristics of the data. During time series processing, it is common that a time series is divided into a large amount of smaller parts named episodes, which are interconnected or partially overlapped [6] and which are important for further processing. For example, interconnected outputs of hydrological models, data collections from traffic monitoring of selected stretches, or long time series divided by segmentation algorithm like Voting experts [7] can be mentioned. Obtained episodes may be processed by a suitable clustering algorithm and divided into the clusters [3, 5].

Various approaches in spheres like recommended systems, decision support systems or tasks based on Case Base Reasoning (CBR) are focused on finding similar sequences (time series episodes) to a sequence entered on the input. In such cases, a suitable cluster of similar sequences is found, which represents the input sequence. Much faster searching is allowed due to finding in set of the cluster representatives which were selected in indexing phase. Thereafter, it is possible to search in depth in a selected cluster or a set of clusters, which are similar to a found episode from the input.

Since each obtained cluster contains a concrete amount of similar episodes, it is suitable to select an appropriate representative, which would describe the whole cluster. Given selected representative is named pattern. Research area aimed to finding patterns, pattern mining, has been studied in several fields. Pattern mining, or pattern recognition, is a scientific discipline focused on object classification into categories or classes [10, 4].

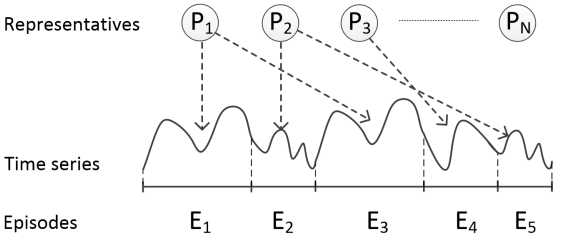


Fig. 1. Collection of Representatives Pointing to Locations in Time Series

Finding the representative of a cluster is defined as finding such set of representative patterns P, which describes episodes E inside these clusters by the most appropriate way. Obtained representatives may be used for the creation of an index file, in which each representative contains a set of pointers to episodes from the base collection (see Figure 1).

Two basic ways for finding representatives are generally known. The first approach is based on selecting one episode, which is the most accurate for a given cluster. The second approach is based on the creation of a representative using the combination of episodes in the cluster. Euclidean distance and other common methods for measuring the similarity between the episodes can be used only while working with the episodes of the identical length. In cases where we have episodes of different lengths, we need a specific algorithm which respects this requirement or an algorithm which is immune to sequence distortions. In the paper, it is described the comparison of the both approaches, and the introduction of an approach which combines the both ways for finding representatives using DTW method is presented (for more details, see Section 2).

The organisation of the paper is following: Dynamic time warping method (DTW) and the utilization of DTW for finding cluster representatives is described in Section 2 and in Section 3. Afterwards, in Section 4, a practical

demonstration of proposed approach is presented. The paper is concluded by Section 5, in which obtained results of suggested approach are discussed and the future work is outlined.

2 Dynamic Time Warping

Recently, finding a signal similar to a signal generated by computers, which consists of accurate time cycles and which achieves a determined finite number of value levels, is a trivial problem. A main attention is focused more likely on the optimisation of searching speed. A non-trivial task occurs while comparing or searching the signals, which are not strictly defined and which have various distortions in time and amplitude. As a typical example, we can mention measurement of functionality of human body (EKG, EEG) or the elements (precipitation, flow rates in riverbeds), in which does not exist an accurate timing for signal generation. Therefore, comparison of such episodes is significantly difficult, and almost excluded while using standard functions for similarity (distance) computation. Examples of such signals are presented in Figure 2a.

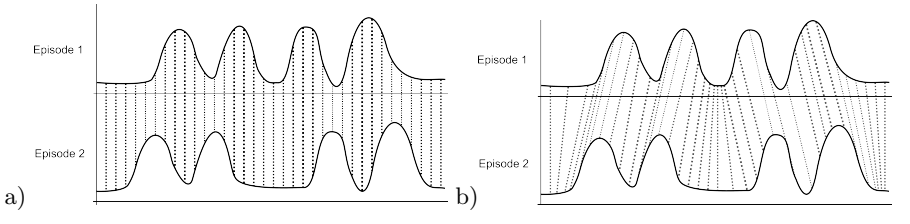


Fig. 2. Standard and DTW Mapping of Episodes

A problem of standard functions for similarity (distance) computation consists in sequential comparison of opposite elements in both episodes (comparison of elements with the identical indexes). Dynamic time warping (DTW) is a technique for finding the optimal matching of two warped episodes using pre-defined rules [1, 9]. Essentially, it is a non-linear mapping of particular elements to match them in the most appropriate way.

The output of such DTW mapping of episodes from Figure 2a can be seen in Figure 2b. This approach was used for example for comparison of two voice patterns during an automatic recognition of voice commands [8].

The main goal of DTW method is a comparison of two time dependent episodes X and Y, where $X = (x_1, x_2, \ldots, x_N)$ is of length $N \in \mathbb{N}$ and $Y = (y_1, y_2, \ldots, y_M)$ is of length $M \in \mathbb{N}$, and to find an optimal mapping of their elements. A detailed description of DTW including particular steps of the algorithm is presented in [1].

3 Using DTW for Finding Cluster Representative

In cases, where it is necessary to gain the most suitable representative of the set of similar episodes, we need to find an algorithm appropriate to a given domain. Sometimes it is possible to use simple average of episodes X and Y, which means that for a representative R is valid, that:

$$R_i = \frac{X_i + Y_i}{2}, \forall i = 1, \ldots, P, \ where \ P = |X| = |Y|. \tag{1}$$

However, this approach is not sufficient in cases, where we have data with distortion. Examples of such episodes are presented in Figure 3a and 3b. If only we used simple average presented in Equation 1, we would achieve an episode showed in Figure 3c. As we can see, this episode absolutely is not a representative and all the information about the episode course is loosed.

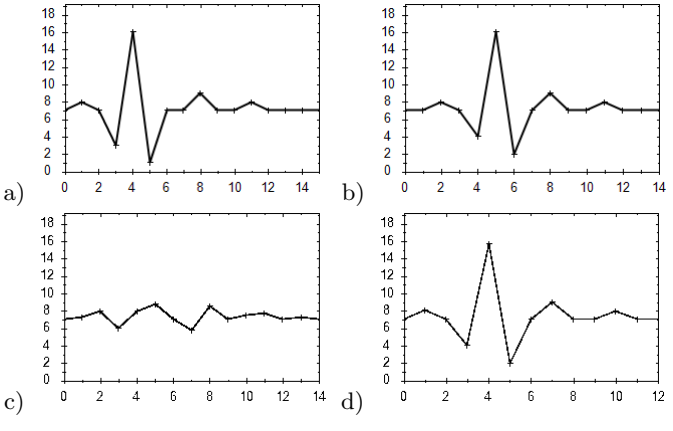


Fig. 3. Similar Episodes X and Y, their Average and Representative Found by DTW

As we can see from Figure 3, it is necessary to find a more appropriate algorithm for domains which yield to distortion. The algorithm should be immune to such distortions. This paper is focused on using DTW for finding a representative of set of similar, but distorted episodes.

3.1 Finding Representative for Episode Couples

The approach for finding a representative of two episodes X and Y by finding the optimal mapping of two episodes using DTW was described in Section 2. In this method, the most important is obtained warped path $p^* = (p_1, \ldots, p_L)$, which allows to find a representative. The approach for finding such representative is described in Algorithm 3.1. The output of presented algorithm applied on episodes in Figure 3 is presented in Figure 3d.

Algorithm 3.1 Searching for Representative from Pair of Episodes

Input: Episodes X and Y
Output: Representative episode R

1. Compute $DTW(X, Y)$ for episodes X and Y; obtain warping path p^*.
2. **Initialization:**
   - R is a representative episode for episodes X and Y.
   - $q = 1$ gives a position in R, $l = 2$ gives a position in warping path p^*.
   - Value in the first position in R is determined as average of values in the first positions of episodes X and Y, e.g. $r_1 = \frac{x_1+y_1}{2}$.
3. **if** $l \leq L$ **then** for couple of the subsequent points of warping path p_l and p_{l-1} perform:
   if $(p_l - p_{l-1}) = (1, 1)$ **then**
   $q = q + 1$;
   A new item $r_q = \frac{x_{n_l}+y_{m_l}}{2}$ is inserted into episode R;
   else if $(p_l - p_{l-1}) = (0, 1)$ or $(p_l - p_{l-1}) = (1, 0)$ **then**
   No item is inserted into representative episodes R;
   end if
   $l = l + 1$
   Repeat Step 3.
   end if
4. Output of the algorithm is representative episode R of length q.

Algorithm 3.1 finds a representative common for two episodes, where both episodes have the same importance. It finds such episode, which is the most similar to the both two episodes. If it is necessary, a one of the episodes may be preferred by adding a weight $w \in (0; \infty)$ and by adjusting a computation of element r_1 and r_q by Equation 2:

$$r_1 = \frac{(x_1 * w) + y_1 * (w - 1)}{w + 1} \; and \; r_q = \frac{(x_{n_l} * w) + y_{m_l} * (w - 1)}{w + 1}. \qquad (2)$$

The impact of adding a weight on achieved representative R for episodes X and Y is following:

- $w = 1$: episodes are equal
- $w \in (1, \infty)$: episode X is preferred
- $w \in (0, 1)$: episode Y is preferred

3.2 Finding Representative for Set of Episodes

Algorithm 3.1 can be applied only on two episodes. However, this is often insufficient in common practice; we need to find a representative for the whole set of episodes in most cases. Given a collection C with generally N episodes,

$C = \{e_1, e_2, \ldots, e_N\}$. The question is, how the presented approach applies on generally N episodes.

A first solution is based on an approach, in which is a representative found step by step by finding particular representatives for episode couples. More precisely, the first step consists of finding representative R_{1-2} for the first two episodes e_1 and e_2. Then, representative R_{1-2-3} is found for a new obtained episode R_{1-2} and for episode e_3. Then, such approach is used for the rest of episodes in the cluster.

However, our experiments showed that this approach is not as much suitable as it could be. It is strongly dependent on the order of particular episodes in collection. The solution is to find an approach that would be immune to the order of elements in an episode. Our proposed approach which solves this problem is presented in Algorithm 3.2.

The presented approach is not restricted only to using DTW as a method for the expression of episode similarity. Of course, DTW could be replaced by any other indicator, for example Euclidean distance or statistical indicators for time series (MAE, MPE, RMSE, etc.). In such cases, it is necessary to adapt steps 2 and 4 of Algorithm 3.2, where instead of finding a representative for the episodes couple by DTW is necessary to use (weighted) average of two compounded episodes. Section 4 describes both two approaches with a visual comparison of the impact to a found representative.

4 Experiments

In this section, a method for determination of similarity between two episodes is presented. Furthermore, the proposed method is compared with other methods. The achieved outputs are visualized with the following structure. The first row of the Figures 4 - 8 consists of episodes, which were used as the input to the algorithm, the second row consists of outputs for the different approaches.

The first output was average of episodes, defined in Equation 1. The second output was from the proposed approach described in Section 3.2. Both outputs are followed by the results using Mean Absolute Error (MAE), and finally as reference, Euclidean distance.

Meaning and usage of DTW method is closer to a human judgement and perception of similarity than a machine definition of physical distance. It is impossible to use a numerical evaluation for the following outputs. The experiments presented in this section were focused on finding such representative, which would describe the characteristics and the important parts of particular episodes.

The first input dataset was a set of similar signals (see Figure 4), which shapes resembled ECG records (described for example in [2]). The signal ended with tiny swings. As we can see from the second row of the episodes in Figure 4, average of values from both episodes absolutely degraded signal information; the shift of signal peaks and drops was smoothed nearly to one level. Also usage of MAE method and Euclidean distance did not provide sufficient results, which

Algorithm 3.2 Searching for Representative from Set of Episodes

Input: Collection C of N episodes
Output: Representative episode R

1. **Initialization:**
   - N is count of input episodes.
   - u is level of collection; $u = 1$.
   - C^1 is the first level of collection; $C^1 = C$.
   - M is count of processed episodes in level u; $M = N - u + 1$.
2. Create from collection C^u, which consists of episodes $\{e_1^u, e_2^u, \ldots, e_M^u\}$, distance matrix $D^u \in \mathbb{R}^{(M \times M)}$, where particular matrix elements are defined as $d_{ij}^u = DTW(e_i^u, e_j^u)$, i.e. matrix elements are created by values of reciprocal mapping of particular episodes.
3. Calculate sum for each row r_i^u in matrix D^u and select a row with the lowest sum value. Find row r_{min}^u, where

$$\sum_{j=1}^{M} d_{min,j}^u = min_{\forall i=1,\ldots M}(\sum_{j=1}^{M} d_{ij}^u)$$

   The found row refers to the episode, which is selected as the most similar to the others in the current collection, and which could be declared as representative R^u of the collection for u-th level.
4. Remove representative R^u from the current collection and create $(N - u)$ new episodes by application of method for searching representative from couple (R^u, e_i^u), described in Section 3.1. This algorithm can be modified by adding weight (preference) to one of the episodes, which can prefer (or discriminate) the importance of the representative R^u.

   if $M > 2$ **then**
   $u = u + 1$;
   $M = M - 1$;
   Repeat from Step 2 for remaining $(N - u)$ episodes;
   else if $M = 2$ **then**
   Select a representative from the two episodes as a representative of the whole original set of episodes C;
   end if

did not differ from average outputs much. On the other way, usage of DTW method for finding representative fully depicted a character of the signal and brought the most accurate results.

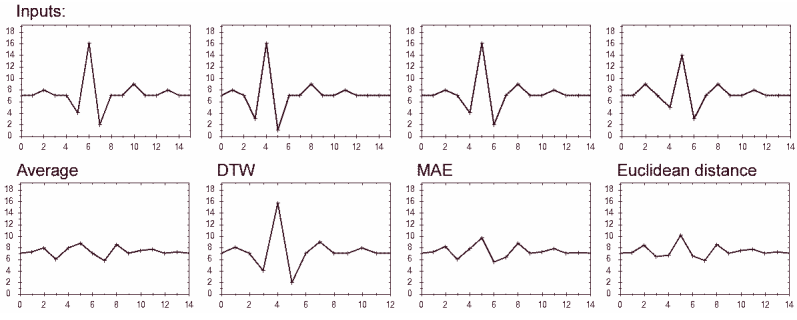


Fig. 4. Experiment with Simplified ECG Signals

The next episode quartet contained signals with the three peaks mutually shifted in time, while each of them had a variable duration (see Figure 5). It was supposed that the representative would have a curve with the three evident peaks. It is obvious from the results, that even though MAE and Euclidean distance worked much better, the loss of information was still noticeable.

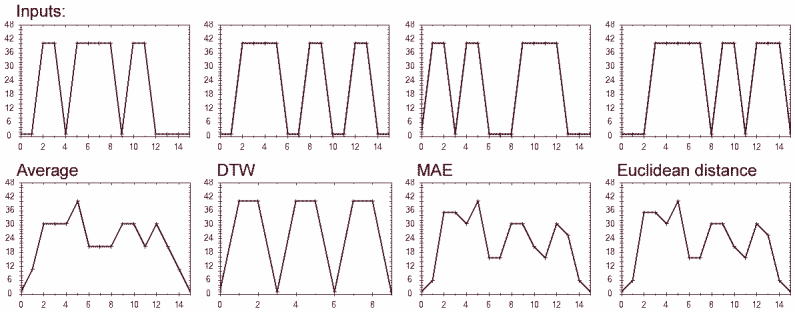


Fig. 5. Experiment with Three Distorted Peeks

The last input dataset represented the situation, in which the signal consisted of two waves - one in a positive and one in a negative part (see Figure 6). These waves were deformed in time, while they were spread or shrunk in X axis. Although the other methods achieved seemingly the best results, the distortion was evident again. The output representative did not contained as high amplitudes as the input waves, did not have smoothed waves and did not detect the constant segments, which were distorted.

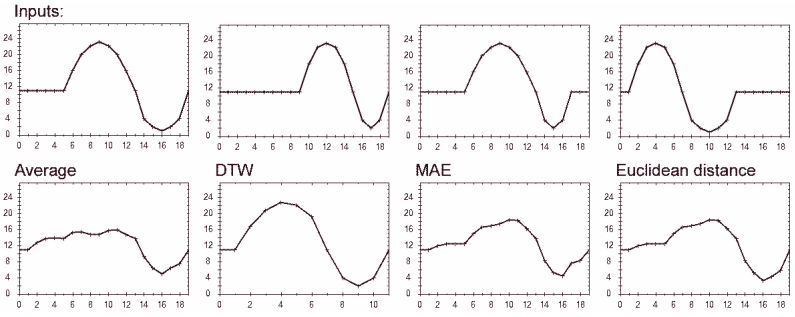


Fig. 6. Experiment with Waves

The most important advantage of the proposed solution is the fact that the Algorithm 3.2 in combination with DTW is able to process even episodes with different lengths. This is very difficult while using other methods, and in some cases even impossible. In these cases it is necessary to shrink the episodes into the identical length, which of course cause the loss of information. Using DTW, we are able to process such episodes with different lengths without any loss of information. In Figures 7 and 8 are presented outputs from proposed algorithm applied on episodes with different lengths.

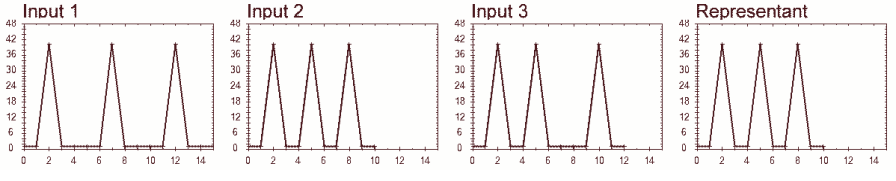


Fig. 7. Set 1 of Episodes with Variable Length

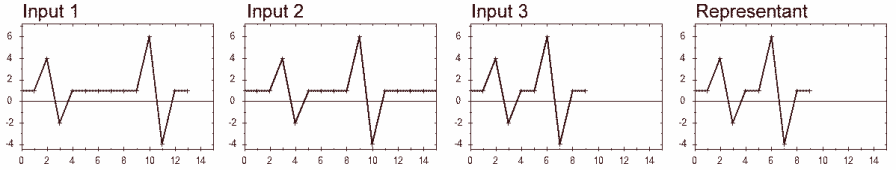


Fig. 8. Set 2 of Episodes with Variable Length

5 Conclusion and Future Work

The real application of proposed algorithm "Searching for Representative from Set of Episodes" described in Section 3.2 showed that it is able to find a representative not only from the set of typical episodes, but also from their distorted variants. The tested input datasets consisted of signals with changed amplitudes and were distorted by time shifting. The proposed solution was compared with conventional methods, in which much worse success was obvious.

Further work will be concentrated on creation of index file, which structure was defined in Section 1, and which visual representation was presented in Figure 1. The aim is to create a sufficiently robust mechanism, which will be able to find all the similar episodes to the selected pattern in data collection during the shortest time. Furthermore, these found episodes will be used for a prediction using the Case-Based Reasoning method. This method requires a suitable mechanism that is able to extract the most similar patterns from the input.

Acknowledgement

This work was supported by the European Regional Development Fund in the IT4Innovations Centre of Excellence project (CZ.1.05/1.1.00/02.0070) and by the SGS, VŠB – Technical University of Ostrava, Czech Republic, under the grant No. SP2013/167 Analysis of Users' Behaviour in Complex Networks.

References

1. Dynamic time warping. In *Information Retrieval for Music and Motion*, pages 69–84. Springer Berlin Heidelberg, Jan. 2007.
2. G. D. Clifford, F. Azuaje, P. McSharry, et al. *Advanced methods and tools for ECG data analysis.* Artech House, 2006.
3. G. Gan, C. Ma, and J. Wu. *Data Clustering: Theory, Algorithms, and Applications.* ASA-SIAM Series on Statistics and Applied Probability. SIAM, MAY 2007.
4. D. J. Hand, P. Smyth, and H. Mannila. *Principles of Data Mining.* MIT Press, Cambridge, MA, USA, 2001.
5. A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: a review. *ACM Computing Surveys*, 31(3):264–323, 1999.
6. E. Keogh, S. Chu, D. Hart, and M. Pazzani. Segmenting time series: A survey and novel approach. *Work*, 57:121, 2004.
7. T. Kocyan, J. Martinovic, M. Podhoranyi, and I. Vondrak. Unsupervised algorithm for retrieving characteristic patterns from time-warped data collections. In *Proceedings of the MAS 2012, The 11th International Conference on Modeling and Applied Simulation*, 2012.
8. L. R. Rabiner and B. B.-H. Juang. *Fundamentals of Speech Recognition.* Prentice Hall, 1993.
9. P. Senin. Dynamic time warping algorithm review. *Information and Computer Science Department University of Hawaii at Manoa Honolulu, USA*, pages 1–23, 2008.
10. S. Theodoridis. *Pattern Recognition.* Elsevier, 3 edition, 2006.

On Updating in XML Peer-to-Peer Databases

Adam Šenk, Michal Valenta

Czech Technical University in Prague, Faculty of Information Technology
Thákurova 9, Prague, Czech Republic
{senkadam,valenta}@fit.cvut.cz

Abstract. Distributed databases are increasingly used in many applications and software systems. Peer-to-Peer databases are interesting part of distributed databases field. This type of distributed database management system allows to connect stand-alone databases via a computer network. Connected databases cooperate on processing of their tasks with other connected databases.
The peer-to-peer databases contains a lot of various distributed and replicated data in various forms. The correct data coordination is the crucial in peer-to-peer databases, it is necessary for correct processing of queries and other database requests. Our work is focused on updates in XML peer-to-peer databases. We provide a review of a peer-to-peer database management system architecture and problems related to implementation of updates. Finally, we consider the requirements on semantic mapping, the part of peer architecture responsible for correct reformulation of global query or update.

Keywords: Peer-to-Peer, XML, Update, Semantic mapping

1 Introduction

Last year, the XML (Extensible Markup Language) celebrated his fifteenth birthday and it has become one of the leading data format. Most of the data published on Web are represented by the XML. The XML is also popular in a data exchange between various systems. XML-native databases use XML as a data format for persistent data storing. Generally speaking, the XML is very popular and variable format and it is used in many systems. Nowadays, most of these systems, that use an XML format, can be seen as big distributed database systems with some limitations. They have many properties that are similar to peer-to-peer database systems properties.

Peer-to-peer database management system (DBMS) is a distributed DBMS with some major differences. It is composed of a large number of database nodes with various bindings. The structure of such system is complicated, changeable and cannot be clearly described by a global schema of distribution. Processing of queries, updates, transactions and other database operations in peer-to-peer DBMS brings new challenges for database developers as they are facing problems caused by the architecture of such systems.

In our work, we investigate an update processing in XML peer-to-peer systems. The problem is that the data stored in the peer-to-peer DBMS are very often interrelated. So it means that updates done on this data could break some dependency or logical structure of stored documents. It is necessary to propagate changes done in one database node to other nodes. We concentrate on basic problem related to update processing in XML peer-to-peer DBMS - identification of data relations in peer-to-peer systems. We analysis what are the biggest problems related to identification of entities, their relations and to coordination of related data updates.

The structure of this paper is the following: Section 2 is summary of related works and known scientific results from field of peer-to-peer DBMS. In Section 3 we define peer-to-peer DBMS and describe architecture and properties of peer-to-peer DBMS. In Section 4 we describe problems related to update processing and to identification of data relations. Section 5 we analyse the requirements on important part of peer-to-peer architecture - the semantic mapping. In Section 6 we discuss the contribution of this work and also the future work of our research.

2 Related Work

There are many woks related to a peer-to-peer databases topic. The book [11] provides a great theoretical background. It reviews the whole distributed databases topic and peer-to-peer databases are described in this context.

A lot of research was done on a field of relational peer-to-peer databases. There exist many distributed peer-to-peer database management systems, one of them is coDB [6]. Greco and Scarcello [7] focused on inconsistencies in query results realized in peer-to-peer databases. Some works are also focused on updates in peer-to-peer databases. Datta [5] and colleagues propose the push/pull strategy for an update processing. Kantere and colleagues introduces distributed triggers for an update processing in relational peer-to-peer databases in [9].

Angela Bonifati and colleagues focused on XML peer-to-peer databases in [2]. They provide a solution for a query reformulation in heterogeneous XML schemas including data and meta-data conflicts. This solution is based on assumption, that each peer knows the structure of data stored in a neighbour peer. Unfortunately, they did not solve updates.

3 Peer-to-Peer Database Managements System

The distributed database management system is a several data processing systems (not necessarily homogeneous) connected via computer network. They communicate and cooperate on data processing. Özsu and Valduriez [11] define a distributed database as a collection of multiple, logically interrelated databases distributed over a computer network. A distributed database management system (DDBMS) is then defined as the software system that permits the management of the distributed database and makes the distribution transparent to the users.

Peer-to-Peer DBMS is distributed system that is composed of stand-alone databases and their bindings. It can be represented by a set of nodes and edges (directed or undirected). Nodes represent some stand-alone DBMS system (SQL database, NoSQL database, distributed database ...) and edges represent bindings between these nodes. The example of XML peer-to-peer system is on Figure 1.

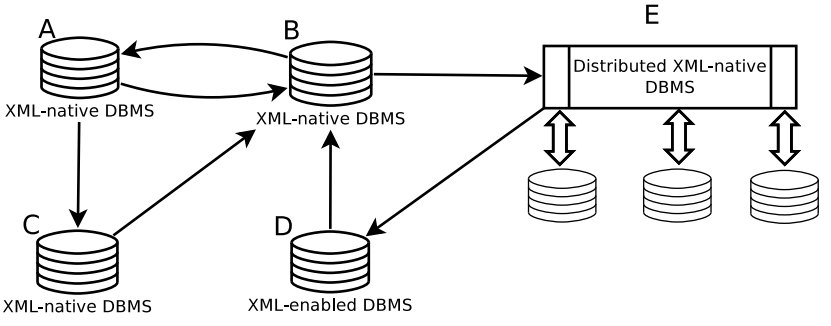


Fig. 1. Example of a schema of some peer-to-peer XML database

There are three significant properties that differ peer-to-peer DBMS from other DDBMS. The first difference is a big amount of nodes in peer-to-peer DBMS. Classical DDBMS with global schema of distribution consist of tens of nodes. Peer-to-Peer DBMS can consists from hundreds or thousands nodes. It causes a massive distribution of data.

The second significant property is a big heterogeneity of such systems and a big autonomy of nodes. Peer-to-Peer DBMS can consists from various database nodes with very various data and data representations.

The autonomy of nodes is the third significant property. The nodes can connect or disconnect any time. It makes the structure of peer-to-peer DBMS very changeable.

3.1 Architecture of Peers

The basic requirements on each DDBMS are as follows:

- **Query processing** - the DBMS must be able to discover all relevant data distributed over the database and efficiently execute the query.
- **Data integration** - the DBMS must be able to discover all related data, even if their representation or schema is different.
- **Data consistency** - the DBMS must maintain the consistency between duplicated data in distributed database.

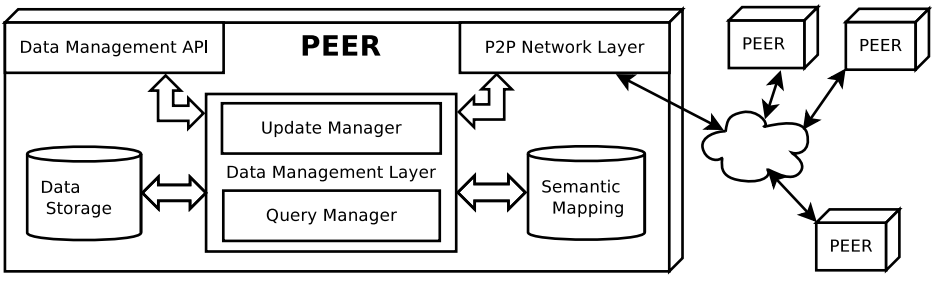


Fig. 2. Peer Architecture

Each peer in a peer-to-peer system must be able to locate and access data distributed and stored in other peers. In Figure 2 is the architecture of a single peer that is to meet this requirements. The architecture consist of four corner stones.

Data management layer is responsible for a query processing. It can have many sub-parts, e.g. the query manager, the update manager and the cache manager. Queries can ask for local data stored in asked peer or for global data distributed in the whole database. Data management layer is also responsible for an execution of remote request send by some other peer.

Data management API is an interface for peer users. The two most common parts are a client containing graphic or text user interface and libraries and public methods for developers. The client allows to submit queries and other requests. It should also provide tools for database tuning and setting its properties. Developers can use application programmable interface to integrate peer functionality into an application.

P2P Network Sublayer is an interface for communication with connected peers over the computer network. It receives requests from the data management layer and distributes them in the whole peer-to-peer DBMS. It also receives messages from other peers and send them to the data management layer to be proceeded.

Persistent storage consists of two basic repositories - *data storage* repository that contains local data and *semantic mapping* that contains meta-information needed for remote query processing. Data management layer uses meta-information for query reformulation when the data stored in the data storage are queried by a remote peer.

Theoretically, the data repository can be replaced by the classic DBMS. Then we see all other layers as a system allowing the connection of such DBMS into a distributed peer-to-peer database.

4 Propagation of Updates

The update in peer-to-peer DBMS is an operation that persistently changes the data stored in some node. The problem of this operation is how to propagate these changes to all nodes in peer-to-peer network. The data in a peer-to-peer database can be replicated in many nodes. It is necessary to distribute the changes over the whole system network. Otherwise, the database consistency will be broken.

To propagate the updates correctly, all interrelated data stored in a peer-to-peer database must be found. The identification of entities, their relations and their states in semi-structured data like XML is a complex problem. We state major problems related to propagation of updates that we are facing.

4.1 Entities in XML

In general, an entity is an existing or real thing. Entity in context of databases is a thing which data can be stored.

The XML is an ordered tree structure consist of edges and nodes. There are three types of node: *element*, *attribute* and *text*. The hierarchical structure of the XML and the three types of nodes in XML tree bring complications to an entity recognition.

An example of two different XML representation of one entity is given in Figures 1.1 and 1.2. As you can see, the structure of an XML fragment is absolutely different in these two cases, but it is obvious that both examples represent the same entity.

Listing 1.1. Book and Authors version 1

```
<book isbn="0-079-13702-4" name="XML Complete" price="49.99
  <author id="1001" firstname="Steven" surname="Holzner" />
<book>
```

Listing 1.2. Book and Authors version 2

```
<book>
  <isbn>0-079-13702-4</isbn>
  <name>XML Complete</name>
  <price>49.99<price>
  <author>
    <id>1001</id>
    <firstname>Steven</firstname>
```

```
    <surname>Holzner</surname>
  </author>
<book>
```

Equality of XML nodes is defined in [8].

Definition 1. *Let name(x), val(x), att(x) and ele(x) be a functions. Function name(x) returns name of given node, function val(x) returns value of given attribute or text node, function att(x) returns list of attribute of given element node and function ele(x) returns the list of descendants of given element node. Two nodes u and v are equal (u = v) if the following conditions are satisfied:*

1. $name(u) = name(v)$
2. *if* u, v *are attribute or text nodes, then* $val(u) = val(v)$
3. *if* u, v *are element nodes, then:*
   (a) *if* $att(u) = a_1, ..., a_m$, *then* $att(v) = a'_1, ..., a'_m$ *and there is a permutation* π *on* $1, ..., m$ *such that* $val(a_i) = val(a_{\pi(i)})$ *for* $i = 1, ..., m$
   (b) *if* $ele(u) = [u_1, ..., u_k]$, *then* $ele(v) = [v_1, ..., v_k]$ *and* $val(u_i) = val(v_i)$ *for* $i = 1, ..., k$

Equality of Entities is not the same as the equality of XML nodes. Two XML nodes can represent one entity even if they are not equal (see Listing 1.1 and Listing 1.2). We define the **entity equality** as follows:

Definition 2. *Two XML nodes u and v are entity equal* ($u =_{ee} v$) *iff they represent the same entity.*

4.2 State of Entity

Each entity can have different states. State is a part of data stored in entity that can differ node by node in one peer-to-peer database. In Listing 1.3 the example is given. This XML code represents the same entity as in two previous example (Listing 1.1 and Listing 1.2), but the value of *price* attribute differs.

Listing 1.3. Book and Authors version 3

```
<book isbn="0-079-13702-4" name="XML Complete" price="56.99
  <author>
    <id>1001</id>
    <firstname>Steven</firstname>
    <surname>Holzner</surname>
  </author>
<book>
```

Attribute *price* is state attribute of an entity and can be different in different nodes. The updates done on parts of XML that represent some state are local and they cannot be distributed over peer-to-peer database.

4.3 Primary Key

The *primary key* is a unique identifier of a data entity. It is the minimal part of an entity which determines its uniqueness. The primary key is used by the data management layer for identifying the updated entity. With primary key defined, it is much more easier to find related data in a distributed peer-to-peer database.

Definition 3. *Lets have two entities* e_1, e_2 *and a primary key function* $PK(e)$ *which returns a primary key and its value for given entity. Then* $PK(e1) = PK(e2) \Leftrightarrow e_1 = e_2$.

The definition 4 of the primary key for relational databases proposed in [4] became a standard for SQL databases. This definition is not usable for a semi-structured hierarchical XML data and it has to be adapted.

Definition 4. *One domain (or combination of domains) of a given relation has values which uniquely identify each element (n-tuple) of that relation. Such a domain (or combination) is called a primary key. In the context of XML hierarchical structure the problem of keys become much more complicated.*

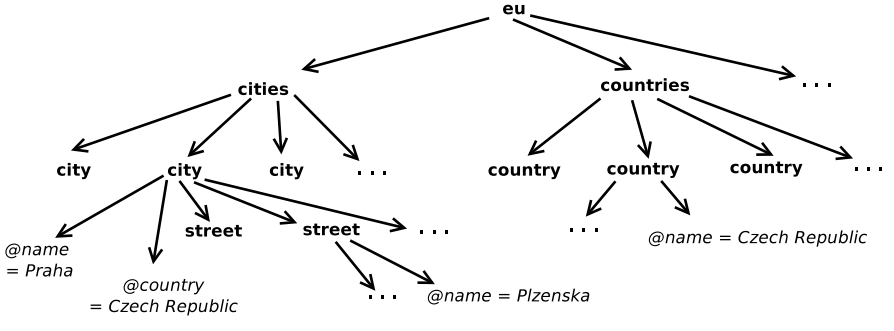


Fig. 3. eu.xml - example of XML data

Let us introduce an example of an XML document in Figure 3 called *eu.xml*. We want to demonstrate problems caused by the hierarchical structure of XML related to the primary key. The *eu.xml* document stores data of member countries in European Union. As you can see, there are three elements with *@name* attribute - **country**,**city** and **street**. The *@name* attribute of **country** element should be unique (there are not two countries with the same name in European Union and there are not two countries with the same name in whole world either). Is the *@name* attribute unique for the **city** element too? Definitely not. There are many examples of cities with the same name (In the Czech Republic, there is one other city that has the same name as the capital city - Prague). There are also examples of cities having the same name as the country (Luxembourg, the member state of European Union has capital city called Luxembourg).

The previous example demonstrates that in an XML data it is necessary to define not only the primary key but also its context. The attribute *@name* is definitely a primary key, but only in context of **countries** elements. That is the reason why keys in XML documents are defined relatively. The definition of *relative keys* was proposed in [3]. We use the XPath language [1] for a key context definition.

Definition 5. *Let P_c and P_t be an XPath expression and let S be a set of XPath expressions. The primary key is defined as $\varphi = P_c(P_t(S))$. If for any node v reached via P_c and for any nodes v_1, v_2 reachable from v via P_t the sets of values $s_1 and s_2$ reached from v_1, v_2 via S satisfies: $s_1 \neq s_2$, then XML document D satisfies φ.*

Example 1. Primary key φ_1 says that *eu.xml* document in Figure 3 is valid if any *@name* attribute of **country** element is unique in a context of **countries** element:

$$\varphi_1 = /eu/countries(/country(@name))$$

5 Semantic Mapping for XML Updates

In chapter 3, we described the architecture of peer-to-peer DBMS. Then we described, in chapter 4, the three major problems that we are facing when we implements updates in XML peer-to-peer DBMS. The analysis done in previous chapters shows the requirements for semantic mapping:

- **Entity recognition** - The DBMS must be able to recognize a single data entity in the hierarchical XML model.
- **Correct update propagation** - The DBMS must to recognize which updates must be propagated in other peers and which not.
- **Common format** - The DBMS must transform XML data to one common format. The data stored in XML can have various formats, so it is necessary to define format that enables the recognition of similarities.

5.1 Definition of Keys

Semantic mapping extracts all important data and provides them to the data management layer. In Section 4.3 we defined the primary key in XML. It is obvious that the semantic mapping must define primary keys of XML data stored in data storage. Primary keys make identification of equal entity easier. Without primary keys is almost impossible to process queries and updates on such complex and heterogeneous data that can be found in peer-to-peer databases.

5.2 Global and Local data

The correct update propagation is one of the requirements on semantic mapping that we state in this section. The data that must be propagated in other peers are called *global* data. The data that are not replicated and are valid in context of single peer are called *local* data. The data management layer cannot automatically recognize the local and global data, this information must be stored in the semantic mapping.

5.3 General model

The most important and also difficult function of the semantic mapping is to map all XML documents to one general model. This model ignores the differences in structure of XML data and allow to compare entities stored in XML format. Developing of such model and developing of mapping on this model is non-trivial task.

6 Conclusion and Future work

In our paper we described the peer-to-peer database management systems. We provided the review of the concept and the description of architecture of the basic unit - peer. Then we concentrated on updates in XML peer-to-peer databases and state the problems to solve when such system is implemented. Finally we state the requirements on the semantic mapping that must be fulfilled in the peer-to-peer database to keep the stored data consistent after updating. Although we do not provide any details we believe that analysis done in our work is a good base for our future research.

Our future work will focus on developing of peer-to-peer DBMS which will allow to connect various independent XML storages in one distributed database. The summarize of information in this paper gives us excellent background. The aim of our future work is to solve processing of updates and other events in distributed system.

References

1. XML Path Language (2013), http://www.w3.org/TR/xpath/
2. Bonifati, A., Chang, E., Ho, T., Lakshmanan, L.V., Pottinger, R., Chung, Y.: Schema mapping and query translation in heterogeneous p2p xml databases. The VLDB Journal 19(2), 231–256 (Apr 2010), http://dx.doi.org/10.1007/s00778-009-0159-9
3. Buneman, P., Davidson, S., Fan, W., Hara, C., Tan, W.C.: Keys for xml. Computer Networks 39(5), 473 – 487 (2002), http://www.sciencedirect.com/science/article/pii/S1389128602002232
4. Codd, E.F.: A relational model of data for large shared data banks. Commun. ACM 13(6), 377–387 (Jun 1970), http://doi.acm.org/10.1145/362384.362685

5. Datta, A., Hauswirth, M., Aberer, K.: Updates in highly unreliable, replicated peer-to-peer systems. In: Distributed Computing Systems, 2003. Proceedings. 23rd International Conference on. pp. 76 – 85 (may 2003)
6. Franconi, E., Kuper, G., Lopatenko, A., Zaihrayeu, I.: Queries and updates in the codb peer to peer database system. In: Proceedings of the Thirtieth international conference on Very large data bases - Volume 30. pp. 1277–1280. VLDB '04, VLDB Endowment (2004), http://dl.acm.org/citation.cfm?id=1316689.1316813
7. Greco, G., Scarcello, F.: On the complexity of computing peer agreements for consistent query answering in peer-to-peer data integration systems. In: Proceedings of the 14th ACM international conference on Information and knowledge management. pp. 36–43. CIKM '05, ACM, New York, NY, USA (2005), http://doi.acm.org/10.1145/1099554.1099564
8. Hartmann, S., Link, S.: Efficient reasoning about a robust xml key fragment. ACM Trans. Database Syst. 34(2), 10:1–10:33 (Jul 2009), http://doi.acm.org/10.1145/1538909.1538912
9. Kantere, V., Manoubi, M., Kiringa, I., Sellis, T., Mylopoulos, J.: Peer coordination through distributed triggers. Proc. VLDB Endow. 3(1-2), 1561–1564 (Sep 2010), http://dl.acm.org/citation.cfm?id=1920841.1921038
10. Masud, M., Kiringa, I.: Transaction processing in a peer to peer database network. Data & Knowledge Engineering 70(4), 307 – 334 (2011), http://www.sciencedirect.com/science/article/pii/S0169023X10001527
11. Özsu, M., Valduriez, P.: Principles of distributed database systems. Springer (2011)

Author Index

Babskova, Alisa, 109
Bača, Radim, 36

Campr, Michal, 80

Dráždilová, Pavla, 109, 129
Dvorský, Jiří, 1, 59, 87

Hlopko, Marcel, 13
Holiš, Michal, 87

Janoška, Zbyněk, 59
Jezek, Karel, 80

Kocyan, Tomáš, 129
Krátký, Michal, 36
Kurš, Jan, 13

Lukáš, Petr, 36

Marek, Lukáš, 1, 26
Martinovič, Jan, 87, 109, 129
Moravec, Pavel, 87

Pászto, Vít, 1, 26
Plaček, Martin, 87
Platoš, Jan, 48, 119

Richta, Karel, 70
Robenek, Daniel, 48

Skřivánek, Zdeněk, 70
Slaninová, Kateřina, 129
Snášel, Václav, 48, 109, 119
Soori, Hussein, 119
Svatoň, Václav, 109

Šenk, Adam, 139

Tuček, Pavel, 1, 26

Valenta, Michal, 139
Vraný, Jan, 13

Zjavka, Ladislav, 98