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 2011 Workshop

DATESO

Databases, Texts
Specifications, and Objects

2011

http://www.cs.vsb.cz/dateso/2011/
http://www.ceur-ws.org/Vol-706/

ARG
AMPHORA RESEARCH GROUP

ČSKI

Supported by

MIR Labs

IEEE
IEEE Systems, Man and Cybernetics Society
CzechoSlovakia

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

April 20 – 22, 2011
Písek

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:	254
Impression:	150
Edition:	1st
First published:	2011

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

Workshop Partner

DATESO 2011 workshop proceedings were published in cooperation with:

SoSIReČR

SOCIAL NETWORK OF COMPUTER SCIENTISTS IN THE REGIONS OF THE CZECH REPUBLIC

SoSIReČR

Social Network of Computer Scientists in the Regions of the Czech Republic

The project aims to promote communication and establish cooperation between the Czech IT communities in academic and corporate sectors. The main output will be a Web portal that will buiild a social network of computer scientists., which will differ from the usual social and professional networks as Facebook or LinkedIn by providing unique tools to search for IT specialists, to create teams (for joint projects, tenders), and in general to allow cooperation. The aim of the project activities is to increase the competitiveness of the Czech Republic in the field of computer science, as well as improvement of the status of IT in the Czech Republic and its contribution to society.

Personal interconnection of educational and research activities and the cooperation of academic and applied research are required for high-quality tertiary education and its fruitful cooperation with industry and Government. From the social network will benefit students and graduates, scientific personnel, schools, and businesses. It will be possible to search here for a job or an interesting project, skilled workers or whole teams, even the schools can compare their learning plans with the situation on the labor market. Beta-version of the portal will be available from April 2011, but now it is possible to test partial functionality (scientific profile) on: http://www.sosirecr.cz/forms/sciprof/.

Another component of the project is to support personal meeting of the IT experts in the regions (both among themselves and with companies), the sharing of experience and the identification of needs. These are the conferences, regional seminars and workshops, aimed at improving communication between the academic, corporate and public sector. One of the following activities are for example Conversations with computer scientists (Hovory s informatiky), organised in the framework of this project by Institute of Computer Science, Academy of Sciences of the CR.

On the SoSIReČR project cooperate:

Faculty of Mathematics and Physics, Charles University in Prague
Institute of Computer Science, Academy of Sciences of the Czech Republic
Czech Technical University in Prague
The Faculty of Informatics and Statistics, University of Economics, Prague
Higher Vocational School and Technical High School, Šumperk

Project (no CZ.07/2.4.00/12.0039) is running in period 01.11.2009 - 31.10.2012.

For more information please visit the **sosirecr** *web pages.*

european social fund in the czech republic
EUROPEAN UNION
MINISTRY OF EDUCATION YOUTH AND SPORTS
INVESTMENTS IN EDUCATION DEVELOPMENT

Steering Committee

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

Program Committee

Václav Snášel (chair)	VŠB-Technical University of Ostrava, Ostrava
Jaroslav Pokorný	Charles University, Prague
Karel Richta	Czech Technical University, Prague
Vojtěch Svátek	University of Economics, Prague
Peter Vojtáš	Charles University, Prague
Dušan Húsek	Inst. of Computer Science, Academy of Sciences, 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

Organizing Committee

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

Preface

DATESO 2011, the international workshop on current trends on Databases, Information Retrieval, Algebraic Specification and Object Oriented Programming, was held on April 20 – 22, 2011 in in OtavArena hotel, Písek.

The 11th 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 the 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 Computer science. The proceedings of DATESO 2011 are also available at DATESO Web site: `http://www.cs.vsb.cz/dateso/2011/` and CEUR Workshop Proceeding site: `http://www.ceur-ws.org/Vol-706/` (ISSN 1613-0073). The Program Committee selected 22 papers (12 full papers, 4 special session full papers and 6 posters) from 33 submissions, based on three independent reviews.

The workshop program also included 2 invited lectures: *Database Trends and Directions: Current Challenges and Opportunities* by George Feuerlicht, and *Simple Mathematical Models in Biometric Image Analysis* by Khalid Saeed.

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 and its proceedings as well as the Czech Society for Cybernetics and Informatics, SoSiReČR project and MIR Labs for their support of publishing this issue.

April, 2011 V. Snášel, J. Pokorný, K. Richta (Eds.)

Table of Contents

Full Papers

Posters

Special Session: Data Processing for Geocomputation

Invited Papers

Best Paper

Top-k Search in Product Catalogues

Martin Šumák and Peter Gurský

Faculty of science, University of Pavol Jozef Šafárik
Jesenná 5, 040 01 Košice, Slovakia martin.sumak@student.upjs.sk,
peter.gursky@upjs.sk

Abstract. In the era of huge datasets, the top-*k* search becomes an effective way to decrease the search time of top-*k* objects. The original top-*k* search requires a monotone combination function and lists of objects ordered by attribute values. Our approach of the top-*k* search is motivated by complex user preferences over product catalogues. Such user preferences are composed of the local user preferences of the attributes' values (user defined arbitrary fuzzy functions, one for each attribute) and a user defined monotone combination function. This paper compares two different approaches of the top-*k* search for this type of non-monotone query. The first approach uses several B+trees, one for each attribute, and it is based on ordered lists. The second approach is new for this type of query and requires an R-tree index.

1 Introduction

The top-*k* search becomes more popular with increasing datasets sizes. Users are usually interested in a few best objects rather than a large list of objects as a result. Top-*k* algorithms usually follow two main goals. First, they minimize the number of source data to be processed. Second, the algorithms try to minimize the number of accesses to the sources and the computation time as well.

Our research in the area of the top-*k* search is motivated by product catalogue search. Users of the common e-catalogues have usually limited options of a preference specification. Typically, the only way to simplify product selection is to order the products by a single attribute. Sometimes, a user can restrict the object set by an interval of attribute values.

Our goal is to enable usage of more complex user preferences. Complex user preferences allow user to specify his top-*k* objects more accurately. On the other hand, preferences should be easy to specify and understandable for common user. Our model of user preferences consists of local preferences to attribute values and a global monotone combination function.

User's local preference to the values in domain D_A of attribute A is represented by a fuzzy function f_A: $D_A \rightarrow [0; 1]$. The fuzzy function gives the value 1 for the attribute values the most preferred by user and the value 0 for the attribute values that the user does not accept. Values between 0 and 1 represent the rates of the user acceptance of the attribute value.

V. Snášel, J. Pokorný, K. Richta (Eds.): Dateso 2011, pp. 1–12, ISBN 978-80-248-2391-1.

Example 1: Imagine a user searching for a cheap laptop with medium screen size. Such preferences can be expressed by fuzzy functions similar to the ones shown on figure 1. We can see that user accepts laptops cheaper than approximately 700 € with a screen size between 11” and 15,5”. Moreover, we know that preferred screen sizes are between 12” and 14” and that the cheaper the laptop is the better preference it has.

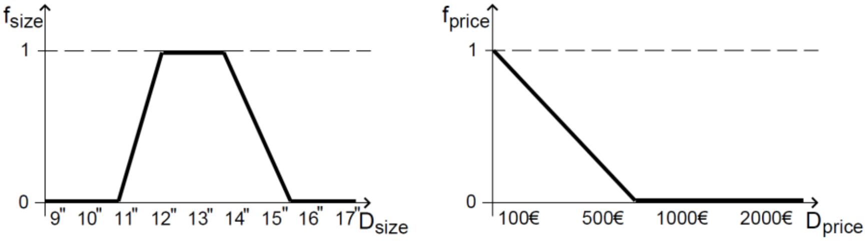


Fig.1. User’s local preferences to the screen sizes and the laptops prices

The fuzzy functions can be specified explicitly by user using sliders [4]. Alternatively they can be learned from objects’ ranking or by a click analysis on the catalogue website [12, 13].

The second part of user preferences is a global monotone combination function that combines the fuzzy values of the local preferences to the overall object value. If user considers m attributes in his preferences, the combination function C: $[0; 1]^m \to \mathbf{R}$ expresses the overall value of the user preferences to the object. The higher the overall value, the more preferred the object is.

Example 2: Let us consider the previous example. Our user can specify that the price is twice as important as the screen size. We can express this importance as weights w_{size}= 1 and w_{price}= 2. Then the combination function can be expressed as weighted sum of the fuzzy values:

$$C(f_{size}(size), f_{price}(price)) = 1*f_{size}(size) + 2*f_{price}(price)$$

The monotone combination function is quite difficult to express by a common user. Therefore the preferred approach uses a predefined combination function (sum, minimum and product) or it can be learned as well as the local preferences [12, 13]. A learned combination function has usually a form of a weighted sum or a set of monotone fuzzy rules [13].

Top-k search algorithms like *Threshold algorithm* (TA), algorithm *No Random Access* (NRA) [1] or *3 phased-No Random Access* (3P-NRA) can be used with user preferences mentioned above [4, 12].

The contribution of this paper is:

- new efficient algorithm for the top-k search based on R-tree (see section 4)
- experimental efficiency comparison of R-tree based algorithm to algorithms TA and 3P-NRA (see section 5)

In [2], algorithms like TA, NRA (or 3P-NRA) are used for searching over several types of attributes i.e. ordinal, metric, hierarchical, etc. Our new solution based on R-tree supports only ordinal attributes so far. Ordinal attributes in TA, NRA (or 3P-NRA) algorithms are handled by B+trees. In this paper we call these algorithms B+trees based approach.

2 Problem definition

For a given set **S** of objects we have to find k most preferred objects for the user. Each object $O \in \mathbf{S}$ has the same m attributes with real values $v_1(O),\ldots, v_m(O)$, where function v_i: $\mathbf{S} \rightarrow \mathbf{R}$ for all $i \in \{1, \ldots, m\}$. Input obtained from the user consists of m fuzzy functions $f_1,\ldots,f_m$ (or less if user does not consider all attributes) and a monotone combination function C. The overall value of object O is $C(f_1(v_1(O)),\ldots, f_m(v_m(O)))$. For example, if C is a weighted sum, user is expected to specify only nonnegative weights – one for each considered attribute. Then we have:

$$C(f_1(v_1(O)),\ldots,f_m(v_m(O))) = w_1{*}f_1(v_1(O)) + \ldots + w_m{*}f_m(v_m(O))$$

where $w_1,\ldots,w_m$ are the weights. The bigger the overall value, the more preferred the object O is to user. Output is a list of k objects from **S** ordered from the most preferred objects to the less preferred ones.

3 B+trees based approach

The original TA [1] and its derivates (NRA, 3P-NRA) require:

- m lists of pairs having form <object identifier, attribute value> previously ordered by attribute values
- a monotone combination function

In the TA, NRA, 3P-NRA algorithms, the ordered lists are read sequentially and the values from the lists are used as an input for a monotone combination function. Typically a top-k algorithm returns top-k objects after processing only a part of the lists.

Our approach requires the monotonicity of a combination function C, having fuzzy values of the local preferences as an input. In the naïve approach, the lists could be sorted according to the local preferences. Then, having the lists ordered by fuzzy values of the local preferences and a monotone combination function, we can use any TA-like algorithm to find top-k objects. Unfortunately, every user usually has different local preferences as well as different combination function. The sorting prior to every top-k search is highly ineffective – it is cheaper to do a table scan and compute the overall values for all objects in **S**.

Instead of ordering the lists, a B+tree can be prepared over each attribute of objects in **S** [2]. The main idea of this approach is that it does not need an ordered list to offer the ordered sorted access.

Example 3. Let us assume that the price and screen size attributes are indexed separately each in one B+tree. According to the fuzzy functions shown on figure 1, i.e. the user local preferences, the B+trees can be traversed to simulate the sorted accesses. The price tree is traversed from a minimal attribute value in ascending direction using pointers between leafs of the B+tree. In the case of the screen size two pointers are created, both starting at the some attribute value with the highest fuzzy value, i.e. 13” with fuzzy value 1. The first pointer traverses leafs of the B+tree in descending direction and the second one in ascending direction (see Fig.2).

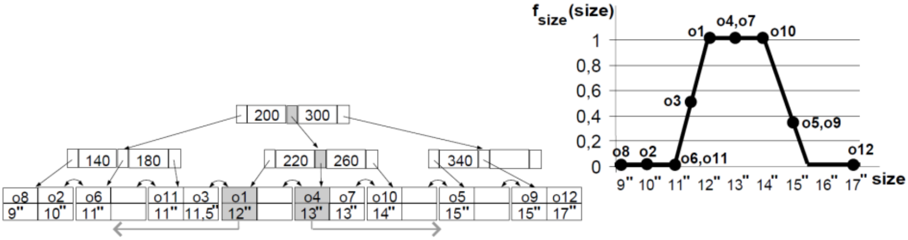


Fig.2. Traversing B+tree organized by attribute screen size during the sorted access simulation for TA-like algorithms

Having simple functions (partially linear), the points to identify the pointers and directions, i.e. the extremes of the fuzzy function, can be found very easily. The entry of a B+tree leaf with the highest fuzzy value can be identified by simple traversing from the root to the appropriate leaf.

At the beginning of the sorted access simulation [2] the entries with the local maximums of a fuzzy function are identified, based on which a set of pointers to leaf entries is obtained. Each pointer points to a leaf entry that should be returned as the next entry in the sorted access simulation. For each pointed entry a fuzzy value is computed. The entry with the highest fuzzy value is returned and its pointer is shifted to the neighbor entry of the corresponding direction.

The sorted access simulation allows us to use any TA-like algorithm. Instead of the original NRA, we prefer to use an improvement of the NRA algorithm called 3P-NRA with significantly better search time as shown in [2] especially in combination with the proposed heuristics. Hence we use 3P-NRA in our experiments in section 5.

4 Searching over R-tree

A contribution of this paper is a top-*k* search algorithm based on R-tree. As shown in the experiments, this approach is much more effective than B+trees based approach.

Since each object O can be represented by the point $p(O) = (v_1(O),\ldots, v_m(O))$ in m-dimensional space (note that p: $\mathbf{S} \rightarrow \mathbf{R}^m$) the set $\mathbf{S}$ of objects can be stored in multidimensional R-tree index [8, 9]. The figure 3 shows an example of R-tree index used for point data. For searching top-*k* objects we adapted algorithm *Incremental Nearest Neighbor* (INN) [10] (also known as *sorted access*) commonly used for searching *k*-nearest neighbors. For formal description of our algorithm we first have to define some concepts.

Definition 1: *Point* P in m-dimensional space is vector $\mathrm{P} = (\mathrm{P}_1,\ldots, \mathrm{P}_m) \in \mathbf{R}^m$.

Remark 1: Having $O \in \mathbf{S}$ and point P such that $\mathrm{P} = p(O)$ then $\mathrm{P}_i = v_i(O)$ for all $i \in \{1,\ldots, m\}$.

Definition 2: *Rectangle* R (parallel with axes) is a pair of points $R = (\mathrm{L}, \mathrm{H})$ where $\mathrm{L}_i \leq \mathrm{H}_i$ for all $i \in \{1,\ldots, m\}$.

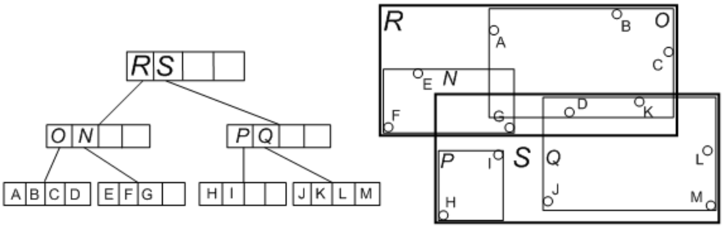


Fig.3. Example of R-tree for two-dimensional point data and capacity of nodes equal to 4

For the implementation and the tests the more effective variant R*-tree [9] was used. R*-tree has the same structure and properties as the R-tree, it differs only in the way of construction. However, the algorithm for k-nearest neighbors and for top-k search is the same for both of them.

Real values of objects' attributes come from different ranges (screen sizes of laptops are from 8" to 17" while the prices are from 400 € to 3000 €) and they are in different units. Therefore we will not build an R-tree index over real values of attributes but we will build it over linear normalized values where all values are within interval [0; 1]. The user defined local preferences (fuzzy functions) will be adjusted according to such normalized data. The normalization is necessary for effective utilization of R*-tree because of R*-tree aims e.g. to have as quadratic nodes as possible.

INN algorithm offers the objects within R-tree in the incremental way and in order from the nearest ones from a query point Q. In top-k search the input is not a single point but it consists of the user's preferences. Objects on the output should be ordered from best for the user. This can be easily achieved by ordering the priority queue within INN algorithm by some other value – not by minimal distance from Q but by maximal overall value (using combination function). For a node **N** we have to compute the maximal possible overall value for any object in a sub-tree rooted by the node **N**. For this purpose we define function h.

Definition 3: Function h: $\mathbf{S} \cup \mathbf{V} \rightarrow \mathbf{R}$, where **V** is a set of nodes of R-tree, is defined as follows: $h(\mathrm{E}) = C(y_1,\ldots, y_m)$ where for all $i \in \{1, \ldots, m\}$ $y_i = f_i(v_i(\mathrm{E}))$ if $\mathrm{E} \in \mathbf{S}$ or $y_i = max\{f_i(x)\text{: } \mathrm{L}_i \leq x \leq \mathrm{H}_i\}$ if $\mathrm{E} \in \mathbf{V}$ and (L, H) is minimal bounding rectangle of E.

Even though user's preferences can be specified by arbitrary fuzzy functions, nevertheless it must be possible to compute the maximum of the fuzzy function on an arbitrary interval and also to compute a value in any permissible x. We prefer to work with the partially linear fuzzy functions. They are easy to define by users using sliders in a graphic interface and they are also simple in computations.

The figure 4 shows a graphical representation of an example of function h. The value $h(O)$ of object O is greater than the value $h(\mathbf{N})$ of node **N**. Object O is therefore better for user than any possible object in a sub-tree of node **N** (any point in rectangle (L, H)).

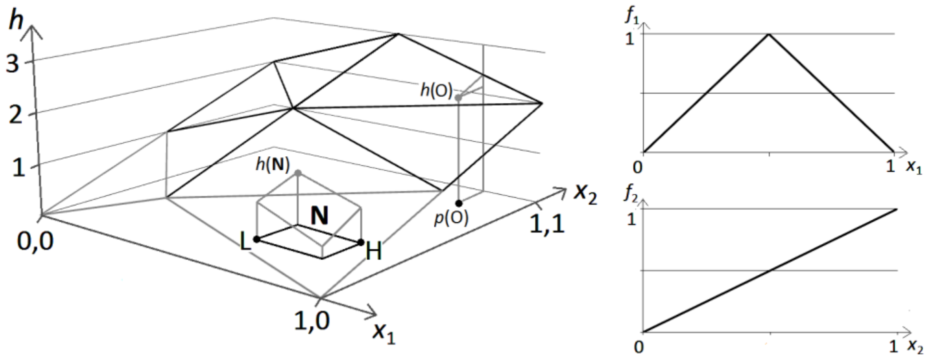


Fig.4. Graphical representation of the function $h(E) = C(y_1, y_2) = y_1 + 2 * y_2$ over normalized data having two attributes of objects with user-defined fuzzy functions f_1 and f_2 on the right

Algorithm preferential top-*k* search over R-tree:

```
Input: R-tree index of objects from S, fuzzy functions
       $f_1,…, f_m$, combination function $C$ and number $k$
Output: ordered list of $k$ objects with the highest value
        of the $h$ function
1. pqueue = empty priority queue ordered by the value of
   the $h$ function of its elements in descending order
2. result = empty list of objects
3. add the root node of the R-tree in the empty pqueue
4. while the result does not contain $k$ objects do
      a. let E be the first element of the pqueue, remove
         E from the pqueue
      b. if E is an inner node then add all its child
         nodes to the pqueue
      c. if E is a leaf node then add all its objects to
         the pqueue
      d. if E is an object then add object E to the end
         of the result
5. return result
```

Algorithm starts with the root node as the only element in the priority queue. At the beginning all the objects are taken into account (root contains all the objects in its sub-tree). The idea of the algorithm is to remove the node at the top of the priority queue, insert all its child nodes instead (or objects if the node was a leaf) with respect to ordering. This is repeated until some object appears on the top of the priority queue. Then the object is put to the end of the result list (the next best object to user). Priority queue ensures that on the top there is an element (i.e. a node or an object) with the maximal value of the function *h* among all elements in the priority queue. Moreover the priority queue can be organized to prefer object to a node with the equal value of function *h*. If the element on the top of the priority queue is a node, then its sub-tree can contain an object with higher value of function h than any other object present in the priority queue. On the other side, if there is an object on the top of the priority queue, its value of function h is higher than or equal to the value of function h of any other object in the priority queue and also of any object in sub-trees of nodes in the priority queue. Hence the first object appeared on the top of the priority queue is the best of all objects. The second object appeared on the top of the priority queue is the best of all remaining objects (i.e. the second best of all objects) and so on.

Inserting all child nodes to the priority queue requires the data (rectangles of child nodes) from the node itself only. Therefore it is possible to store only pointers to nodes in the priority queue for better efficiency and load real nodes when the information about their child nodes or objects is needed.

5 Tests and results

In the tests the time and number of IOs of the following three algorithms are compared: TA, 3P-NRA and the *preferential top-k search over R-tree*. Since R*-tree is the most efficient variant of R-tree, we used it in all the following tests. The utilization of R*-tree nodes was within the range from 30 to 90. In the tests we used three different distributions of artificial random data: Gaussian, exponential and uniform. We used sets of 100 000 and 1000 000 objects. The combination function was weighted sum where weights were chosen randomly from interval [1; 5]. Fuzzy functions for the evaluation of objects were chosen also randomly from 4 types (familiarly called: ascending, descending, hill, valley) used in [21]. All the tests were done on computer with 2GB of RAM, 2core Intel Centrino processor and SSD hard disk. The purpose of the tests was to reveal the influence of the data distribution, user preferences or the number of considered attributes on time and number of IOs. Presented values are the average values of 5 identical tests.

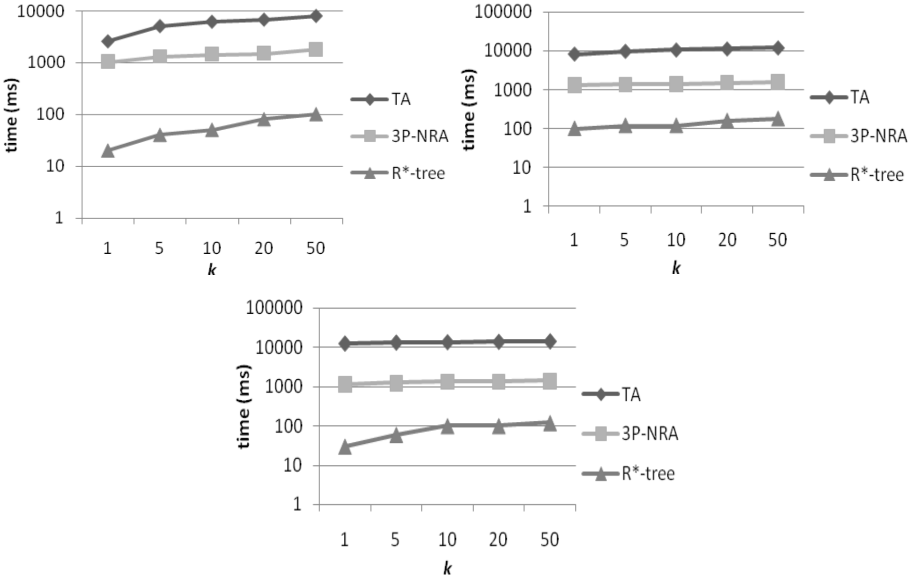


Fig.5. Search time of the top-1, 5, 10, 20, 50 objects in ms over 100 000 random objects with 10 attributes of exponential (top-left), Gaussian (top-right) and uniform (bottom) distribution with all 10 attributes in a query

First set of charts (figure 5) describes the search time of top-*k* objects. We can observe significant speedup of searching when data are indexed in R*-tree and the query uses all 10 attributes of objects. The dependence of number of IOs copies the time dependence for all three algorithms.

The question is whether the R*-tree based algorithm will be so fast if the query would contain only some of all indexed attributes. TA and 3P-NRA algorithms can be considered faster trivially (because they read fewer ordered lists). The R*-tree based algorithm always searches over the R*-tree structure containing all the attributes independently of the user query.

Next set of charts (figures 6, 7) describes the search time of the top-10 and the top-50 objects of 10 attributes with less than 10 attributes used in a query.

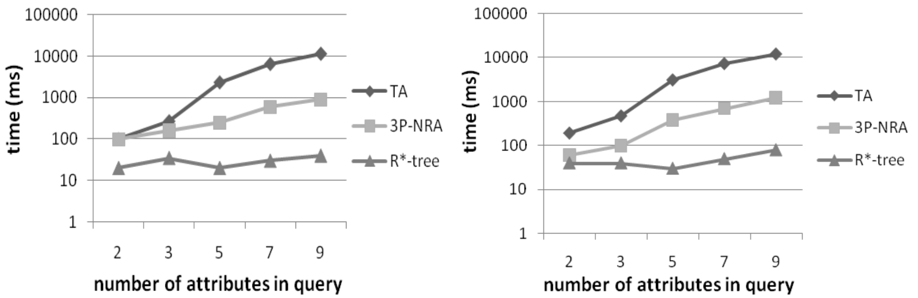


Fig.6. Search time of the top-10 (left) and the top-50 (right) objects in ms over 100 000 random objects of 10 attributes with uniform distribution with 2, 3, 5, 7, 9 attributes in a query

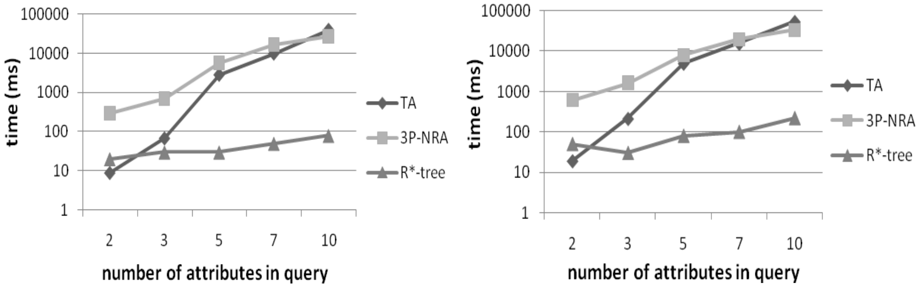


Fig.7. Search time of the top-10 (left) and the top-50 (right) objects in ms over 1 000 000 random objects of 10 attributes with exponential distribution with 2, 3, 5, 7, 10 attributes in a query

As we can see on the charts above the R*-tree based algorithm is fast even for a small number of attributes in the query. In farther tests we observe that this property of R*-tree based algorithm is preserved also for a bigger set of objects (1 000 000), more attributes (20) and also for other distributions (exponential, Gaussian).

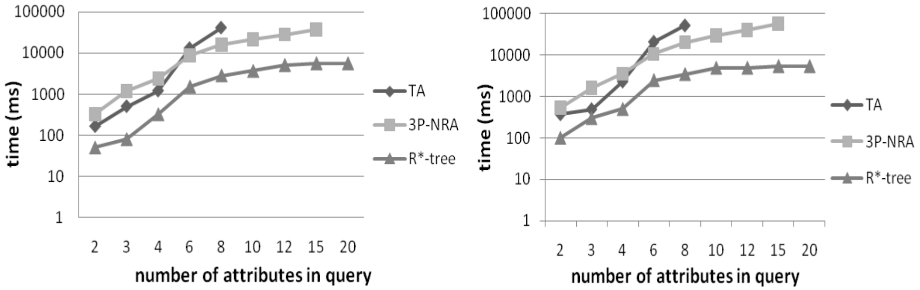


Fig.8. Search time of the top-10 (left) and the top-50 (right) objects in ms over 1 000 000 random objects of 20 attributes with Gaussian distribution with 2, 3, 4, 6, 8, 10, 12, 15, 20 attributes in a query

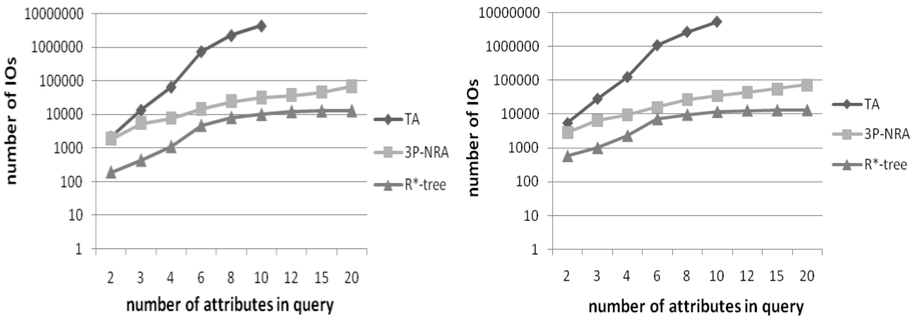


Fig.9. Number of IOs in the search of the top-10 (left) and the top-20 (right) objects over 1 000 000 random objects of 20 attributes with Gaussian distribution with 2, 3, 4, 6, 8, 10, 12, 15, 20 attributes in a query

The tests show that R*-tree based approach is much faster than the B+trees based approach. The remarkable speedup is reached by decreasing the number of IOs – from several B+trees to one R*-tree.

6 Related work

This paper studies the top-*k* search for complex user preferences composed of arbitrary local preferences and a monotone combination function. Local preferences and combination function together generate a non-monotone function giving the overall value.

The area of the top-*k* search was extensively researched in the last 7-11 years. The main stream of the top-*k* search algorithms [1, 14, 15, 16, 17], we call them TA-like algorithms, considers having a monotone combination function and possibly distributed ordered lists for each attribute.

The query composed of local preferences and monotone combination function was introduced in [2]. As shown in section 3 the simulation of sorted access to the lists allows us to use TA-like algorithms.

A special branch of top-*k* algorithms is considered to be embedded in RDBMS [18, 19, 20]. These approaches are concerned with augmenting the query optimizer to consider rank-joins during plan evaluation. Optimization can be effective especially in the case of very selective attributes. The rank-join algorithms require ordered data on input similarly to TA-like algorithms. The way of ordering is not considered or the ordering of attribute domains is used implicitly.

The approaches in [5, 6] do the top-*k* search with an arbitrary (also non-monotone) query analyzing the aggregation function with mathematical methods. If a ranking function analysis over any domain sub-region is possible (to find the maximum and possibly recognize monotonicity), according to authors this approach is able to find the top-*k* objects in an effective way. In our opinion this analysis is rather difficult to be done for an arbitrary function. However, we could not proceed in further analysis of this approach because the source codes or a deeper description of the algorithms are not available.

R-tree is designed especially for multidimensional range queries and similarity search (nearest neighbor queries). Moreover R-tree can be effectively used for top-*k* queries. The idea of using R-tree for the top-*k* search is briefly presented in [7] where authors compare the top-*k* search over R-tree and over their new index called "Ranked Join Index". They consider only a simple weighted sum as a top-*k* query. Note that our top-*k* query, consisting of m fuzzy functions $f_1,\ldots,f_m$ and a monotone combination function C, is more complex. The composition of fuzzy functions and a combination function is not monotone in general, thus our approach has a higher expressive power. Since the "Ranked Join Index" requires a monotone top-*k* query, it cannot be used with our query.

7 Conclusions and future work

Unlike separate ordered lists in the original TA-like algorithms our approach uses single R-tree or R*-tree. Since R-tree is a multidimensional index, each attribute can be represented as one dimension. A disadvantage in comparison to the original TA-like algorithms is that we must know the values of all objects' attributes and all the data must be stored locally in one structure. We must have the whole R-tree prepared prior to the query (we emphasize that we have to prepare the R*-tree only once, not prior to each query). We assume that a condition of locally accessible data prepared in advance is performable for almost all attributes. Nevertheless in the case of dynamic, remote or not ordinal attributes, the TA-like algorithms and the R-tree based approach can be combined. Since algorithm for the top-*k* search over R-tree can be used to produce an ordered list, we can carry out the top-*k* search using a TA-like algorithm. One ordered list for a TA-like algorithm can be simulated by the top-*k* search algorithm over R-tree and several (not so many) ordered lists can be obtained in other way, i.e. simulation over remote B+tree. In this case we can dramatically decrease the number of ordered lists read by a TA-like algorithm and consequently the overall time of the top-k search as well. This is the idea of our future work.

Acknowledgement

This work was partially supported by VEGA 1/0131/09 and by the Agency of the Slovak Ministry of Education for the Structural Funds of the EU, under project ITMS: 26220120007.

References

1. R. Fagin, A. Lotem, M. Naor, "Optimal Aggregation Algorithms for Middleware", in Proc. ACM PODS, 2001
2. P. Gurský, R. Pázman, P. Vojtáš, "On supporting wide range of attribute types for top-k search", *Computing and Informatics*, Vol. 28, no. 4, 2009, ISSN 1335-9150, p. 483-513..

3. T. Horváth, “A Model of User Preference Learning for Content-Based Recommender Systems”, *Computing and informatics* Vol. 28 (2009), No. 4: SAV, Slovakia, 2009, ISSN 1335-9150, p: 453-481.
4. V. Vaneková, P. Vojtáš, “Fuzziness as a Model of User Preference in Semantic Web Search”, in Proc. IFSA-EUSFLAT 2009, pp. 998-1003
5. Z. Zhang, S. Hwang, K. Chang, M. Wang, C. Lang, Y. Chang, “Boolean + Ranking: Querying a Database by K-Constrained Optimization,” In SIGMOD 2006.
6. D. Xin, J. Han, K. Chang, “Progressive and Selective Merge: Computing Top-K with Ad-Hoc Ranking Functions,” in SIGMOD 2007.
7. P. Tsaparas, T. Palpanas, Y. Kotidis, N. Koudas, D. Srivastava, “Ranked Join Indices”, ICDE, pp.277, 2003
8. A. Guttman, “R-Trees: A Dynamic Index Structure for Spatial Searching”, SIGMOD Conference 1984
9. N. Beckmann, H.P. Kriegel, R. Schneider, B. Seeger, “The R*-Tree: An Efficient and Robust Access Method for Points and Rectangles”, SIGMOD Conference 1990: 322-331
10. G. R. Hjaltason , H. Samet, “Distance browsing in spatial databases”, ACM Transactions on Database Systems (TODS), v.24 n.2, p.265-318, June 1999
11. G. R. Hjaltason, H. Samet, “Ranking in Spatial Databases”, Proceedings of the 4th International Symposium on Advances in Spatial Databases, p.83-95, August 06-09, 1995
12. P. Gurský, T. Horváth, R. Novotný, V. Vaneková, P. Vojtáš, “UPRE: User preference based search system”, In IEEE/WIC/ACM Web Inteligence (2006)
13. T. Horváth, P. Vojtáš, “Ordinal Classification with Monotonicity Constraints”, In Proc. 6th Industrial Conference on Data Mining ICDM (2006)
14. R. Akbarinia, E. Pacitti, P. Valduriez, “Best Position Algorithms for Top-k Queries”, in VLDB, 2007.
15. H. Bast, D.Majumdar, R. Schenkel, M. Theobald, G. Weikum, “IOTop-k: Index-Access Optimized Top-k Query Processing”. In VLDB, 2006.
16. W. Balke, U. Güntzer, “Multi-Objective Query Processing for Database Systems”, in VLDB, 2004.
17. U. Güntzer, W. Balke, W. Kiessling, “Towards Efficient Multi-Feature Queries in Heterogeneous Enviroments”, in ITCC, 2001.
18. F. Ilyas, W. Aref, A. Elmagarmid, “Supporting Top-k Join Queries in Relational Database”, in VLDB, 2003.
19. F. Ilyas, R. Shah, W.G. Aref, J. S. Vitter, A.K. Elmagarmid, “Rank-Aware Query Optimization”, in SIGMOD, 2004.
20. C. Li, K. Chang, I. F. Ilyas, S. Song, “RankSQL: Query Algebra and Optimization for Relational Top-k Queries”, in SIGMOD, 2005.
21. P. Gurský, “Towards better semantics in the multifeature querying”, proceedings of Dateso 2006, ISBN 80-248-1025-5, pages 63-73, 2006

XML Schema Integration with Reusable Schema Parts*

Jakub Klímek, Jakub Malý, and Martin Nečaský

XML Research Group, Department of Software Engineering
Faculty of Mathematics and Physics, Charles University in Prague
Malostranské náměstí 25, 118 00 Praha 1, The Czech Republic
{klimek, maly, necasky}@ksi.mff.cuni.cz

Abstract. Modern information systems may exploit numerous XML formats for communication. Each message may have its own XML format for data representation which causes problems with integration and evolution of their schemas. Manual integration and management of evolution of the XML formats may be very hard. We tackled this problem in our previous work, however, for simplicity reasons, we omitted the possibility of exploiting reusable schema parts. In this paper, we complement our previous work with additional methods for schema integration which exploit reusable schema parts that quite often appear in XML schemas. This further helps a domain expert to get a precise mapping to a conceptual diagram, which then integrates the XML formats and facilitates their evolution - a change that is made once in the conceptual diagram is propagated to the XML formats.

Keywords: XML schema, conceptual modeling, reverse-engineering, integration

1 Introduction

Today, XML is a standard for communication in various information systems like web services, etc. A web service provides an interface composed of several operations. The structure of incoming and outgoing messages is described in a form of XML schemas. If the XML schemas of communicating web services differ, the problem of their integration comes to the scene. When the XML schemas are integrated, another problem arises. Since the business evolves in time the XML schemas need to be adapted too.

We aim at the problem of integration of XML schemas by mapping them to a common conceptual schema. In our previous work [5, 10, 4], we have introduced a framework for XML schema integration and evolution. It supposes a set of XML schemas that are conceptually related to the same problem domain. As a problem domain, we can consider, e.g., purchasing products. Sample XML schemas may be XML schemas for purchase orders, product catalogue, customer detail, etc. The central part of the framework is a conceptual schema of the problem domain. Each XML schema is then mapped to the conceptual schema. In other words, the conceptual diagram integrates the XML schemas. We then exploit the mappings to evolve the XML schemas when a

* This work was supported in part by the Czech Science Foundation (GAČR), grant number P202/11/P455 and in part by the grant SVV-2011-263312.

change occurs. Simply speaking, the change is made only once at the conceptual level and then propagated to the affected XML schemas. It is also possible to exploit the mappings to derive interfaces of semantic web services described in SAWSDL as we show in [11, 8]. In [7, 4] we have introduced a method of XML schema integration, which will be briefly introduced later.

Contributions In practice, a conceptual diagram and XML schemas exist separately, i.e. there are no mappings between both levels. This disallows to exploit the integration and evolution capabilities of our framework. In our work [9], we have introduced a method for deriving required XML schemas from the conceptual diagram and in [7] and [4] we have described a reversed method for mapping of an existing XML schema to the conceptual diagram. In this paper, we extend this method by utilizing inheritance constructs that often appear in XML schemas and that are supported by our conceptual model to get even better results and more comfortable way of integrating them.

Our aim is not to develop new methods for measuring schema similarities. These methods have been already intensively studied in the literature. Instead, we exploit the existing ones and combine them together. For this, we provide an algorithm skeleton that can be supplemented by various similarity methods. An important contribution of the method, not considered by existing similarity methods, is an active participation of a domain expert. This is necessary, since we need to achieve exact mapping.

Outline The rest of the paper is organized as follows. In Section 2, we present related work. In Section 3, we briefly present a simplified version of our conceptual model for XML. Section 4 briefly describes the algorithm from [7, 4] which assists a domain expert during mapping discovery and we enhance it with methods for dealing with inheritance. In Section 5, we evaluate the presented approach. Finally, Section 6 concludes.

2 Related work

Recent literature (surveyed in [6]) has been focused on a discovery of mappings of XML formats to a common model. We can identify several motivations. XML schemas are hardly readable and a friendly graphical notation is necessary. This motivation has appeared in [2][3] or [15]. A survey of these approaches can be found in [17]. They introduce an algorithm for automatic conversion of a given XML schema to a UML class diagram. The result exactly corresponds to the given XML schema. However, these approaches can not be applied in our case – we need to map an XML schema to an existing conceptual diagram. There are also approaches aimed at an integration of a set of XML format into a common XML format. These works include, e.g. the DIXSE framework [13] or Xyleme project [12]. Approaches that convert or map XML formats to ontologies are DTD2OWL [14], which presents a simple method of automatic translation of an XML format with an XML schema expressed in DTD into an ontology. More advanced methods are presented in [1] and [16]. They both introduce an algorithm that automatically maps an XML format to an ontology. This is close to our approach since a conceptual diagram can be understood as an ontology. In both cases, the domain expert can edit the discovered mappings but is not involved in the discovery process directly. For a more detailed description of related work see [7].

3 Our Conceptual Model

In this section, we will introduce our conceptual model for XML. We follow the Model-Driven Architecture (MDA) principle which is based on modeling data at several levels of abstraction. The most abstract level contains a conceptual schema of the problem domain. The language applied to express the conceptual schema is called *platform-independent model (PIM)*. The level below is the *platform-specific level* which specifies how the whole or a part of the PIM schema is represented in a particular platform. In our case, the platform is XML.

3.1 Platform-Independent Model

A PIM schema is based on UML class diagrams and models real-world concepts and relationships between them. It contains three types of components: classes, attributes and associations.

Definition 1. *Let $\mathcal{L}$ be a set of string labels and $\mathcal{D}$ be a set of datatypes. A* schema in the platform independent model (PIM schema) *is a 9-tuple $\mathcal{S} = (\mathcal{S}_c, \mathcal{S}_a, \mathcal{S}_r, \mathcal{S}_e, name, type, class, participant, card)$, where:*

- *$\mathcal{S}_c$ and $\mathcal{S}_a$ are sets of* classes *and* attributes *in $\mathcal{S}$, respectively.*
- *$\mathcal{S}_r$ is a set of* binary associations *in $\mathcal{S}$. $\mathcal{S}_e$ is a set of* association ends *in $\mathcal{S}$. A binary association is a set $R = \{E_1, E_2\}$, where E_1, $E_2 \in \mathcal{S}_e$ and $E_1 \neq E_2$. For any two associations R_1, $R_2 \in \mathcal{S}_r$ it must hold that $R_1 \cap R_2 \neq \emptyset \Rightarrow R_1 = R_2$. In other words, no two associations share the same end.*
- *$name : \mathcal{S}_c \cup \mathcal{S}_a \rightarrow \mathcal{L}$ resp. $name : \mathcal{S}_r \rightarrow \mathcal{L} \cup \{\lambda\}$ assigns a* name *to each class, attribute and association. $name(R) = \lambda$ means that $R \in \mathcal{S}_r$ does not have a name.*
- *$type : \mathcal{S}_a \rightarrow \mathcal{D}$ assigns a data type to each attribute.*
- *$class : \mathcal{S}_a \rightarrow \mathcal{S}_c$ assigns a class to each attribute. For $A \in \mathcal{S}_a$, we will say that A* is an attribute *of $class(A)$ or A* belongs to *$class(A)$.*
- *$participant : \mathcal{S}_e \rightarrow \mathcal{S}_c$ assigns a class to each association end. For $R = \{E_1, E_2\} \in \mathcal{S}_r$, we will say that $participant(E_1)$ and $participant(E_2)$* are participants of *R or that they* are connected by *R.*
- *$card : (\mathcal{S}_a \cup \mathcal{S}_e) \rightarrow \mathcal{C}$ assigns a cardinality to each attribute and association end.*

The members of $\mathcal{S}_c$, $\mathcal{S}_a$, and $\mathcal{S}_r$ are called components *of $\mathcal{S}$.*

We display PIM schemas as UML class diagrams. A class is displayed as a box with its name at the top and attributes at the bottom. An attribute is displayed as a pair comprising the attribute name and cardinality. The data type is omitted to make the diagram easy to read. An association is displayed as a line connecting participating classes with the association name and cardinalities.

For a given association $R = (E_1, E_2)$, we will often use notation (C_1, C_2) as an equivalent of $(participant(E_1), participant(E_2))$ if there are no more associations connecting C_1 and C_2. We will also need a construct called a PIM path.

Definition 2. *A* PIM path *P is an ordered sequence $\langle R_1, \ldots, R_n \rangle$ of associations from $\mathcal{S}_r$, where $(\forall i \in \{1, n\})((R_i) = (C_{i-1}, C_i))$. C_0 and C_n are called* start *and* end *of P. Functions $start$ and end return for P the start and end of P, respectively.*

An example of a PIM path in our sample PIM depicted in Figure 1(a) is $Path = \langle (Purchase, Item), (Item, Product), (Product, Supply)\rangle$. *Purchase* and *Supply* are start and end of the PIM path, respectively.

3.2 Platform–Specific Model

The *platform-specific model (PSM)* enables to specify how a part of the reality is represented in a particular XML schema in a UML-style way. We introduce it formally in Definition 3. We view a PSM schema in two perspectives. From the *grammatical perspective*, it models XML elements and attributes. From the *conceptual perspective*, it delimits the represented part of the reality. Its advantage is clear – the designer works in a UML-style way which is more comfortable then editing the XML schema.

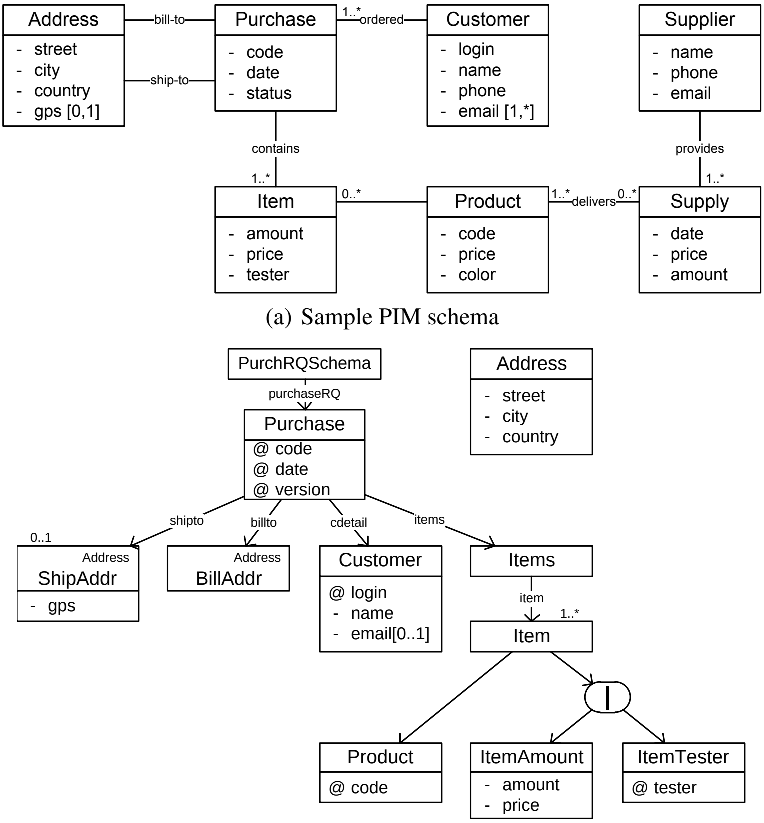
Definition 3. *Let $\mathcal{L}$ be a set of string labels and $\mathcal{D}$ be a set of datatypes. A* PSM schema *is a* 16*-tuple* $\mathcal{S}' = (\mathcal{S}'_c, \mathcal{S}'_a, \mathcal{S}'_r, \mathcal{S}'_e, \mathcal{S}'_m, C'_{\mathcal{S}'}, name', type', class', xform', participant', card', cmtype', attributes', content', repr')$*, where*

- $\mathcal{S}'_c$*,* $\mathcal{S}'_a$*, and* $\mathcal{S}'_m$ *are sets of* classes, attributes*, and* content models *in* $\mathcal{S}'$*, respectively.*
- $\mathcal{S}'_r$ *is a set of* directed binary associations *in* $\mathcal{S}'$*.* $\mathcal{S}'_e$ *is a set of* association ends *in* $\mathcal{S}'$*. A directed binary association is a pair* $R' = (E'_1, E'_2)$*, where* E'_1*,* $E'_2 \in \mathcal{S}'_e$ *and* $E'_1 \neq E'_2$*. For any two associations* R'_1*,* $R'_2 \in \mathcal{S}'_r$ *it must hold that* $R'_1 \cap R'_2 \neq \emptyset \Rightarrow R'_1 = R'_2$*.*
- $C'_{\mathcal{S}'} \in \mathcal{S}'_c$ *is a class called* schema class *of* $\mathcal{S}'$*.*
- $name' : \mathcal{S}'_c \cup \mathcal{S}'_a \rightarrow \mathcal{L}$ *resp.* $name' : \mathcal{S}'_r \rightarrow \mathcal{L} \cup \{\lambda\}$ *assigns a* name *to each class, attribute and association.*
- $type' : \mathcal{S}'_a \rightarrow \mathcal{D}$ *assigns a data type to each attribute.*
- $class' : \mathcal{S}'_a \rightarrow \mathcal{S}'_c$ *assigns a class to each attribute. For* $A' \in \mathcal{S}'_a$*, we will say that* A' is an attribute *of* $class'(A')$ *or* A' belongs to $class'(A')$.
- $participant' : \mathcal{S}'_e \rightarrow \mathcal{S}'_c \cup \mathcal{S}'_m$ *assigns a class or content model to each association end. For* $R' = (E'_1, E'_2)$*, where* $X'_1 = participant'(E'_1)$ *and* $X'_2 = participant'(E'_2)$*, we call* X'_1 *and* X'_2 parent *and* child *of* R'*, respectively. We will also sometimes call both* X'_1 *and* X'_2 participants *of* R' *and say that* X'_1 *is the* parent *of* X'_2 *and* X'_2 *is a* child *of* X'_1*, denoted* $parent'(R')$ *and* $child'(R')$*, respectively.*
- $xform' : \mathcal{S}'_a \rightarrow \{\mathtt{e}, \mathtt{a}\}$ *assigns an* XML form *to each attribute. It specifies the XML representation of an attribute using an XML element declaration with a simple content or an XML attribute declaration, respectively.*
- $card' : \mathcal{S}'_a \cup \mathcal{S}'_e \rightarrow \mathcal{C}$ *assigns a* cardinality *to each attribute and association end.*
- $cmtype' : \mathcal{S}'_m \rightarrow \{\mathtt{sequence}, \mathtt{choice}, \mathtt{set}\}$ *assigns a* content model type *to each content model. We distinguish 3 types: sequence, choice and set, respectively.*
- $attributes' : \mathcal{S}'_c \rightarrow 2^{(\mathcal{S}'_a)}$ *assigns an ordered sequence of distinct attributes to each class* C'*. It must hold that* $A' \in attributes'(C') \Leftrightarrow C' = class'(A')$*.*
- $content' : \mathcal{S}'_c \cup \mathcal{S}'_m \rightarrow 2^{(\mathcal{S}'_r)}$ *assigns an ordered sequence of distinct associations to each class or content model* X'*. It must hold that* $R' \in content'(X') \Leftrightarrow X'$*is the parent of* R'*.*

- *$repr' : \mathcal{S}'_c \setminus \{C'_{\mathcal{S}'}\} \rightarrow \mathcal{S}'_c \setminus \{C'_{\mathcal{S}'}\}$ assigns a class $\overline{C'}$ to another class C'. C' is called* structural representative *of $\overline{C'}$. It must hold that $C' \notin repr'(\overline{C'})$. Neither C', nor $\overline{C'}$ can be the schema class.*

The graph $(\mathcal{S}'_c \cup \mathcal{S}'_m, \mathcal{S}'_r)$ with classes and content models as nodes and associations as directed edges must be a directed forest with one of its trees rooted in the schema class $C'_{\mathcal{S}'}$. Members of $\mathcal{S}'_c$, $\mathcal{S}'_a$, $\mathcal{S}'_r$, and $\mathcal{S}'_m$ are called components *of $\mathcal{S}'$.*

A sample PSM schema is depicted in Figure 1(b). As can be seen from the definition, PSM introduces similar constructs to PIM: classes, attributes and associations.



(a) Sample PIM schema

(b) Sample PSM schema

Fig. 1. Sample PIM and PSM schemas

The PSM-specific constructs have precisely defined semantics. Briefly, a class models a complex content. The complex content is specified by the attributes of the class and associations in its content (their ordering is given by functions $attributes'$ and $content'$). An attribute models an XML element declaration with a simple content or

XML attribute declaration depending on its XML form (function $xform'$). An association models an XML element declaration with a complex content if it has a name. Otherwise, it models only that the complex content modeled by its child is nested in the complex content modeled by its parent. If a class C' is a structural representative of another class $repr'(C')$, the complex content modeled by C' extends the complex content modeled by $repr'(C')$. This is exactly our definition of a *reusable schema part* as multiple PSM classes can be structural representatives of another target PSM class, meaning that all of them reuse the definition of the target PSM class.

The PSM schema represents a part of a PIM schema. A class, attribute or association in the PSM schema may be mapped to a class, attribute or association in the PIM schema. In other words, there is a mapping which specifies the semantics of classes, attributes and associations of the PSM schema in terms of the PIM schema. The mapping must meet certain conditions to ensure consistency between PIM schemas and the specified semantics of the PSM schema. This mapping is called *interpretation of the PSM schema against the PIM schema.*

Definition 4. *Let* $R = \{E_1, E_2\} \in \mathcal{S}_r$ *be an association. An* ordered image *of* R *is a pair* $R^{E_1} = (E_1, E_2)$ *(or* $R^{E_2} = (E_1, E_2)$*).*

We will use $\overrightarrow{\mathcal{S}_r}$ to denote the set of all ordered images of associations of $\mathcal{S}'$, i.e. $\overrightarrow{\mathcal{S}_r} = \bigcup_{R \in \mathcal{S}'_r} \{R^{E_1}, R^{E_2}\}$. We need these definitions to be able to distinguish direction of PIM association, which is normally not needed in PIM.

Definition 5. *An* interpretation *of a PSM schema* $\mathcal{S}'$ *against a PIM schema* $\mathcal{S}$ *is a partial function* $I : (\mathcal{S}'_c \cup \mathcal{S}'_a \cup \mathcal{S}'_r) \rightarrow (\mathcal{S}_c \cup \mathcal{S}_a \cup \overrightarrow{\mathcal{S}_r})$ *which maps a class, attribute or association from* $\mathcal{S}'$ *to a class, attribute or ordered image of an association from* $\mathcal{S}$*, respectively. For* $X' \in (\mathcal{S}'_c \cup \mathcal{S}'_a \cup \mathcal{S}'_r)$*, we call* $I(X')$ interpretation of X'. $I(X') = \lambda$ *denotes that* I *is not defined for* X'*. In that case, we will also say that* X' does not have an interpretation.

Let a function $context'_I : \mathcal{S}'_c \cup \mathcal{S}'_a \cup \mathcal{S}'_r \cup \mathcal{S}'_m \rightarrow \mathcal{S}'_c$ *return for a given component* X' *of* $\mathcal{S}'$ *the closest ancestor class to* X' *on* $path'(X')$ *so that* $I(context'_I(X')) \neq \lambda$*. The following conditions must be satisfied:*

$$I(C'_{\mathcal{S}'}) = \lambda \tag{1}$$

$$(\forall C' \in \mathcal{S}'_c \text{ s.t. } repr'(C') \neq \lambda)(I(C') = I(repr'(C'))) \tag{2}$$

$$(\forall A' \in \mathcal{S}'_a \text{ s.t. } I(A') \neq \lambda)(class(I(A')) = I(context'_I(A'))) \tag{3}$$

$$(\forall R' \in \mathcal{S}'_r \text{ s.t. } I(child'(R')) = \lambda)(I(R') = \lambda) \tag{4}$$

$$(\forall R' \in \mathcal{S}'_r \text{ s.t. } I(child'(R')) \neq \lambda)(I(R') = (I(context'_I(R')), I(child'(R')))) \tag{5}$$

Each PSM class, attribute or association can have an *interpretation* against a component of the PIM schema. This mapping means that in the PSM schema the particular PSM component models the concept represented by the target PIM component in the PIM schema.

Note that in the context of mapping of a PSM schema to a PIM schema (interpretation construction), the content models present in a PSM schema are irrelevant as they do not influence the semantics of PSM classes, attributes nor associations.

4 Algorithm

In this section we will enhance our interpretation reconstruction algorithm first introduced in [7] and extended to a framework in [4] so that it takes into account for reusable schema parts. These are represented in our conceptual model as *structural representants*. Because of lack of space in this paper, we will omit some details of the basic algorithm, which can be found in [7, 4].

The algorithm builds an interpretation I of a PSM schema against a PIM schema. I must be correct, it must fulfil Definition 5. Moreover, it must be correct in the conceptual sense, i.e. a PSM component and its PIM interpretation must conceptually correspond to the same real-world concept. We ensure the formal correctness. The conceptual correctness is ensured by a domain expert.

4.1 Overview

The basic algorithm works in three phases. Firstly, it measures initial similarities between PSM and PIM attributes and classes. Secondly, it creates an *initial interpretation* of PSM classes, whose initial similarity to some PIM class is higher than a given threshold. Becasue this is done automatically, there is a possibility that this initial interpretation is not correct. Therefore, it has to be verified by a domain expert. Nevertheless, the initial interpretation usually helps to avoid confirming lots of obvious mapping matches because the domain expert just needs to confirm a list of pre-mapped classes (or uncheck the incorrect ones). The confirmed initial interpretation now becomes a final interpretation and the algorithm moves to its third phase. It builds interpretation of the unmapped PSM classes with an assistance of a domain expert.

We will suppose a PSM schema $\mathcal{S}'$ and a PIM schema $\mathcal{S}$ on the input. The output of the algorithm is an interpretation I of $\mathcal{S}'$ against $\mathcal{S}$. We will enhance parts of the algorithm where the knowledge of reusable schema parts (structural representants) can help. But first, let us motivate a definiton. Let C' be a structural representant of C''. Due to condition 2 of Definition 5, the following must hold: $I(C') = I(C'')$. This means that both C' and C'' need to have the same interpretation in the PIM schema (or both must remain uninterpreted). This also means (from condition 3 of Definition 5 and from the definition of $\mathit{context}'(C')$), that PSM attributes of C' and C'' can only have attributes of the same PIM class as interpretation. Intuitively, C' and C'' represent the same concept in the PSM schema and we can suppose that their names also refer to the same concept. Note that the same goes for every PSM class C''', that would be a structural representant of C'. This justifies the follwing definition.

Definition 6. *Let a function $ss' : \mathcal{S}'_c \rightarrow 2^{(\mathcal{S}'_c)}$ return for each PSM class C' a set of PSM classes, which are (transitively) related to C' by the* structural representative *(repr') relation.*

For example, let C'_1, C'_2, C'_3 and C'_4 be PSM classes. Let $\mathit{repr}'(C'_1) = \lambda$, $\mathit{repr}'(C'_2) = C'_1$, $\mathit{repr}'(C'_3) = C'_1$ and $\mathit{repr}'(C'_4) = \lambda$. Then $ss'(C'_1) = ss'(C'_2) = ss'(C'_3) = \{C'_1, C'_2, C'_3\}$ and $ss'(C'_4) = \emptyset$.

4.2 Measuring Initial Similarity

Attributes. Firstly, the algorithm measures a similarity for each pair of one PIM and one PSM attribute. This is based on their names and datatypes. This phase is not affected by the structural representants and we can skip the detailed description. Suffice to say that results of initial attribute similarity are used in function $S^{init-attrs}(C', C)$ below, which gives us similarity of a PSM class and a PIM class based on their attributes. **Classes.** Let $(C', C) \in \mathcal{C}' \times \mathcal{C}$. The similarity between C' and C is customizable, in this paper it is a weighted sum

$$S^{init-class}(C', C) = w^{init-class} * S^{init-attrs}(C', C)$$
$$+ (1 - w^{init-class}) * max\{S^{str}(name'(C'), name(C)), S^{str}(xml'(C'), name(C))\}$$

where $w^{init-class} \in (0, 1)$ is a weighting factor and $xml'(C')$ is a name of the parent association of C' if any exists. $S^{init-attrs}(C', C)$ is defined as $S^{init-attrs}(C', C) = \sum_{A' \in attributes'(C')} max_{A \in attributes(C)} (S^{init-attr}(A', A))$, i.e. it finds for each PSM attribute $A' \in attributes'(C')$ the most similar PIM attribute A of C and summarizes these similarities.

This is the first place where we can exploit structural representants. For a PSM class C', we can take attibutes of every $C'' \in ss'(C')$, because if those classes have an interpretation, it is the same PIM class for all of them (and similarly for the attributes). Therefore, we define function $attrs_{sr} : \mathcal{S}'_c \rightarrow 2^{(\mathcal{S}'_a)} = \cup_{C'_i \in ss'(C')} attributes'(C'_i)$ and we can redefine:

$$S^{init-attrs}(C', C) = \sum_{A' \in attrs_{sr}(C')} max_{A \in attributes(C)} (S^{init-attr}(A', A))$$

4.3 Initial interpretation

The initial class interpretations are set according to the initial class similarities precomputed in the previous step. It is a simple procedure that takes the most similar pairs of PSM and PIM classes (with similarity greater than a given threshold) and sets these pairs as initial interpretations. Here is another place were we exploit structural representants. Because of the fact that all PSM classes of $ss'(C')$ need to have the same interpretaion (or none at all), when we initially interpret one of them, we can as well initially interpret all of them and the interpretation will be the same PIM class. And, of course, due to the possibility that this interpretation is incorrect, we can provide the user with the comfort of accepting/rejecting the whole group at once. If the domain expert chose to consider structural representatives in both the attribute similarity and the name similarity, this is an effect of the previous modification. The reason for this is that all of the classes from the group will have the same initial similarities, because when we computed the initial similarities for one class from the group, we included all the other classes as well. If, however, the domain expert chose to ignore structural representants at some stage, the similarities will be different and this adjustment may come in handy.

4.4 Final Interpretation

The third part of the algorithm iteratively traverses the PSM classes in $\mathcal{S}'_c$ in pre-order and helps the domain expert to build the final interpretation. Individual steps are shown in Algorithm 1. For an actual PSM class $C' \in \mathcal{S}'_c$, the algorithm firstly constructs $I(C')$ (lines 2 - 6) and also sets the interpretation for all the PSM classes of $ss'(C')$ (lines 7 - 9). This is because all of them must have the same interpretation. Secondly, the algorithm constructs $I(A')$ for each $A' \in attributes(C')$ (lines 10 - 22). Finally, it constructs $I(R')$ for each $R' \in content(C')$ (lines 23 - 25). It can be shown that this algorithm runs in $O(N^3)$ where N is the number of PSM classes and in $O(n \times log(n))$ where n is the number of PIM classes.

Algorithm 1 Interpretation Construction Algorithm

```
 1: for all $C' \in \mathcal{S}'_c$ in post-order do
 2:   for all $C \in \mathcal{S}_c$ do
 3:     $S^{class}(C', C) \leftarrow w^{class} * S^{init-class}(C', C) +$
                  $(1 - w^{class}) * \frac{1}{S^{adj-class}(C', C)}$
 4:   end for
 5:   Offer the list of PIM classes sorted by $S^{class}$ to the domain expert.
 6:   $I(C') \leftarrow C$ where $C \in \mathcal{S}_c$ is the PIM class selected by the domain expert.
 7:   for all $C'' \in ss'(C')$ do
 8:     $I(C'') \leftarrow C$ {here we set the interpretation for the whole group of PSM classes}
 9:   end for
10:   for all $A' \in attributes(C')$ do
11:     for all $A \in \mathcal{S}_a$ do
12:       $S^{attr}(A', A) \leftarrow w^{attr} * S^{init-attr}(A', A) +$
                  $(1 - w^{attr}) * \frac{1}{\mu(I(C'), class(A)) + 1}$
13:     end for
14:     Offer the list of PIM attributes sorted by $S^{attr}$ to the domain expert.
15:     $I(A') \leftarrow A$ where $A \in \mathcal{S}'_a$ is the PIM attribute depicted by the domain expert.
16:     if $I(class'(A')) \neq class(A)$ then
17:       Create PSM class $D' \in \mathcal{S}'_c$; $I(D') \leftarrow class(A)$
18:       Put $A'$ to $attributes'(D')$
19:       Create PSM association $R' = (C', D') \in \mathcal{S}'_r$
20:       Put $R'$ at the beginning of $content'(C')$.
21:     end if
22:   end for
23:   for all $R' \in content'(C')$ do
24:     $I(R') \leftarrow P$ where $P$ is the PIM path connecting $I(C')$ and $I(child'(R'))$ s.t.
        $\mu(I(C'), I(child'(R')))$ is minimal.
25:   end for
26: end for
```

Class Interpretation To construct $I(C')$, the algorithm firstly computes $S^{class}(C', C)$ for each $C \in \mathcal{S}'_c$ at line 3. It is a weighted sum of two similarities. The former is the initial similarity $S^{init-class}(C', C)$. The other is a reversed class similarity adjustment $S^{adj-class}(C', C)$ which we discuss in a while. The algorithm then sorts the PIM classes by their similarity with C' and offers the sorted list to the domain expert at line 5. The expert selects a PIM class from the list and the algorithm sets $I(C')$ to this selected class at line 6.

Class similarity adjustment $S^{adj-class}(C', C)$ is computed on the base of the completed part of I, which includes confirmed initial interpretation. $S^{adj-class}(C', C)$ is a combination of distances between C and PIM classes D_i which are interpretations of the interpreted neighbors of C'. $\mu(C, D)$ is the distance between PIM classes C and D.

Note that Algorithm 1 is a skeleton which needs to be supplemented with methods for (1) measuring distances between PIM classes, (2) combining distances, and (3) selecting candidates for C' structural similarity adjustment. In this paper, we use basic methods to show that the general idea works. For measuring the distance between two PIM classes C and D, we use the length of the shortest PIM path connecting C and D. As the distance combination method, which results in the aimed $S^{adj-class}(C', C)$, we can also choose from various possibilities. In this paper, we use

$$S^{adj-class}(C', C) = (\sum_{i=1}^{n} \frac{\mu(C, I(D'_i))}{n}) + 1$$

where D'_1, …, D'_n are the selected interpreted neighbors of C'. $S^{adj-class}(C', C)$ is the average of the lengths of the shortest PIM paths between C and each $I(D'_i)$.

Finally, we need to decide which mapped neighbors of C' will be selected to compute $S^{adj-class}(C', C)$. We can choose among children of C' or previous siblings of C', as these were already interpreted by the domain expert in this part of the algorithm. Because we have some PSM classes interpreted via the *initial interpretation*, we can use them as another candidates for structural similarity adjustment, if they are close enough. Therefore, we can also select interpreted following siblings, interpreted parent or interpreted ancestors as candidates for structural similarity adjustment. These options are described and experimented with in [4].

Here is another moment where we can exploit reusable schema parts in a form of structural representants. As we choose which interpreted neighbors of C' to use for the structural similarity adjustment, we can also work with the same type of interpreted neighbors of all classes of the group $ss'(C')$. The reasons are the same, because the interpretation of all classes of the group must be the same PIM class.

The rest of the algorithm remains unaffected by the structural representatives, so we describe it only briefly. For details, see [7, 4]. When all the PSM classes have been interpreted or the domain expert decided they should remain uninterpreted, a similar process is performed for PSM attributes of the classes. The possibilities of mapping a PSM attribute in this situation are limited due to the rules that the interpretation must adhere to (see Definition 5). Finally, PSM associations are interpreted with respect to the same rules.

5 Evaluation

In this section, we briefly evaluate the effect of structural representants on building interpretations of PSM classes. For more detailed experiments with the overall method see [7, 4]. We have implemented the introduced method in our tool XCase[1] which was primarily intended for designing XML schemas from a created PIM schema.

[1] `http://xcase.codeplex.com`

Let us suppose an actual PSM class C'. Let the domain expert set $I(C')$ to a PIM class C either when asked or when confirming the initial interpretation. We measure the precision of the algorithm from two points of view. Firstly, we measure the position of C in the list of PIM classes offered to the expert sorted by their S^{class}. We call this precision a *global precision* $\mathcal{P}_G$:

$$\mathcal{P}_G = ((\sum_{C' \in \mathcal{S}'_c} 1 - \frac{order(C) - 1}{n})/n') * 100$$

where n denotes the size of $\mathcal{S}_c$, n' denotes the size of $\mathcal{S}'_c$, and $order(C)$ denotes the order of C in the list. If there are more PIM classes with the same similarity to C', $order(C)$ is the order of the last one. $\mathcal{P}_G = 0$ (resp. 1) if for each PSM class C', the selected PIM class was the last (resp. first).

The global precision is not sufficient. When C is the first class, there can be other PIM classes before C which have their similarity to C' close to $S^{class}(C', C)$ and make it harder to distinguish whether C is or is not a good match for C'. We therefore propose another metric called *local precision* which measures the amount of PIM classes with their similarity to C' close to $S^{class}(C', I(C'))$. It is defined as

$$\mathcal{P}_L = ((\sum_{C' \in \mathcal{S}'_c} 1 - \frac{close(C) - 1}{n})/n') * 100$$

where $close(C)$ denotes the number of PIM classes with their similarity to C' close to $S^{class}(C', C)$. The term *close similarity* can be defined in various ways. In this paper, we say that y is *close* to x if $y \in (x - 0.1, x + 0.1)$.

Intuitively, the effect of using structural representants is a reduction of the number of mapping offers the domain expert needs to go through. This is because when an interpretation of a PSM class C' is constructed, it is automatically constructed for all PSM classes $ss'(C')$ and the domain expert no longer needs to create the interpretation for each one of them.

Additionally, the use of structural representants for class similarity computations may help with global and local precisions. This is, however, dependent on the texts present in the source PSM schema, its structure and selected methods of similarity measurements, which so far can not be determined automatically. Therefore, the experimental results are very complex and their description would not fit into this article.

6 Conclusion

In this paper, we studied the effect of exploiting reusable schema parts on techniques used for mapping of XML formats to a conceptual diagram. We briefly described our basic algorithm from [7, 4] which allows to exploit various similarity measurement methods. Then we introduced our enhancements that allow us to take advantage of the reusable schema parts, which are expressed as structural representants in our conceptual model. Finally, we have provided a biref evaluation of the proposed method.

References

1. R. dos Santos Mello and C. A. Heuser. A Bottom-Up Approach for Integration of XML Sources. In *Workshop on Information Integration on the Web*, pages 118–124, 2001.
2. J. Fong, S. K. Cheung, and H. Shiu. The XML Tree Model - toward an XML conceptual schema reversed from XML Schema Definition. *Data Knowl. Eng.*, 64(3):624–661, 2008.
3. M. R. Jensen, T. H. Møller, and T. B. Pedersen. Converting XML Data to UML Diagrams For Conceptual Data Integration. In *In Proceedings of DIWeb01*, 2001.
4. J. Klímek, I. Mlýnková, and M. Nečaský. A Framework for XML Schema Integration via Conceptual Model. In *Advances in Web, Intelligent, Cloud, and Mobile Systems Engineering - WISE 2010 Symposium and Workshops*. Springer, 2011.
5. J. Klímek and M. Nečaský. Integration and Evolution of XML Data via Common Data Model. In *Proceedings of the 2010 EDBT/ICDT Workshops, Lausanne, Switzerland, March 22-26, 2010*, New York, NY, USA, 2010. ACM.
6. J. Klímek and M. Nečaský. Reverse-engineering of XML Schemas: A Survey. In J. Pokorný, V. Snásel, and K. Richta, editors, *DATESO*, volume 567 of *CEUR Workshop Proceedings*, pages 96–107. CEUR-WS.org, 2010.
7. J. Klímek and M. Nečaský. Semi-automatic Integration of Web Service Interfaces. In *IEEE International Conference on Web Services (ICWS 2010)*, pages 307–314. IEEE Computer Society, 2010.
8. J. Klímek and M. Nečaský. Generating Lowering and Lifting Schema Mappings for Semantic Web Services. In *25th IEEE International Conference on Advanced Information Networking and Applications Workshops, WAINA 2010, Biopolis, Singapore, 22-25 March 2011*. IEEE Computer Society, 2011.
9. M. Nečaský. *Conceptual Modeling for XML*, volume 99 of *Dissertations in Database and Information Systems Series*. IOS Press/AKA Verlag, January 2009.
10. M. Nečaský and I. Mlýnková. On Different Perspectives of XML Schema Evolution. In *FlexDBIST'09*, Linz, Austria, 2009. IEEE.
11. M. Nečaský and J. Pokorný. Designing Semantic Web Services Using Conceptual Model. In *Proceedings of SAC'08*, pages 2243–2247. ACM, 2008.
12. C. Reynaud, J.-P. Sirot, and D. Vodislav. Semantic integration of xml heterogeneous data sources. In *In Proceedings of IDEAS '01*, pages 199–208, Washington, DC, USA, 2001. IEEE Computer Society.
13. P. Rodríguez-Gianolli and J. Mylopoulos. A Semantic Approach to XML-based Data Integration. In *ER '01: Proceedings of the 20th International Conference on Conceptual Modeling*, pages 117–132, London, UK, 2001. Springer-Verlag.
14. P. T. T. Thuy, Y.-K. Lee, and S. Lee. DTD2OWL: Automatic Transforming XML Documents into OWL Ontology. In *In Proceedings of ICIS '09*, pages 125–131, New York, NY, USA, 2009. ACM.
15. Y. Weidong, G. Ning, and S. Baile. Reverse Engineering XML. *Computer and Computational Sciences, International Multi-Symposiums on*, 2:447–454, 2006.
16. L. Xiao, L. Zhang, G. Huang, and B. Shi. Automatic Mapping from XML Documents to Ontologies. In *CIT '04: Proceedings of the The Fourth International Conference on Computer and Information Technology*, pages 321–325, Washington, DC, USA, 2004. IEEE Computer Society.
17. A. Yu and R. Steele. An Overview of Research on Reverse Engineering XML Schemas into UML Diagrams. In *ICITA (2)*, pages 772–777. IEEE Computer Society, 2005.

Transformation of Binary Relationships with Particular Multiplicity*

Zdeněk Rybola[1] and Karel Richta[2]

[1]Department of Software Engineering, Faculty of Information Technology, Czech Technical University in Prague
[2]Department of Computer Science and Engineering, Faculty of Electrical Engineering, Czech Technical University in Prague
[2]Department of Software Engineering, Faculty of Mathematics and Physics, Charles University in Prague
richta@ksi.mff.cuni.cz

Abstract. The paper deals with one small step in the process of model driven development (MDD) or model driven architecture (MDA) – widely used terms nowadays. MDD defines techniques to develop software systems using variety of models together with a set of transformations. MDD specifies several levels of models depending on abstraction – ranging from computation independent models (CIM) to platform independent models (PIM) into platform specific models (PSM), and implementation specific models (ISM). Many CASE tools provide automated support for generating more specific models from more abstract ones – e.g. PSM or ISM from PIM. This task is referred to as forward engineering. Many tools also provide automated support for generating more abstract models from more specific ones – e.g. PIM from PSM or ISM. This task is referred to as reverse engineering. Some tools also support round-trip engineering, which involves both forward and reverse engineering steps such that software artifacts (model and code) become synchronized. However, these tools need exact transformation rules to be defined for such transformations. This paper describes basic principles and restrictions for transformations of binary relationships and transformations of binary relationships with the particular multiplicity from PIM level into PSM level. The idea is illustrated on examples.

1 Introduction

Model driven development is the dream of OMG (Object Management Group), and the top desire is so called round-trip engineering. It is the combination of a forward process (generating code from model) and a reverse process (generating model from code). The necessary equipment for such round-trip process are transformations, which serve as the description of partial steps in this process. OMG offers Unified Modeling Language (UML) for the description of models.

* This work has been partially supported by the grant project of the Czech Grant Agency (GAČR) No. GA201/09/0990 and by SGS grant.

The necessary part of such notation is a language for description of model constraints – in the case of UML it is Object Constraint Language (OCL) – one of the basic parts of UML.

OMG defines MDA or MDD [6] as the set of modeling levels or views. These models can be transformed – from the upper level to a lower one, or from the lower level to an upper one, or between models on the same level of abstraction – some sort of model refactoring. Transformations have to be described precisely in some formal fashion.

OCL [3,10,7] is a specification language. As a part of UML standard [5,4], it is used to define restrictions for the model in UML by constructs such as invariants, pre- and post-conditions connected to model elements which give context for the constraints.

This paper deals with the basic principles of transformations of binary relationships in PIM level into PSM level. To illustrate these principles we suppose the relational database system as the platform, so the SQL serves as the platform specific language. Where particular restrictions are required for transformation of binary relationship to PSM, OCL constraints in PIM level are defined first, so these can be used in transformations to other PSMs or by automated transformation tools such as [9,2,13]. These tools provide support to OCL syntax checking, constraints evaluation for system snapshots and even generating of source code for connected model and constraints.

The paper is structured as follows. Section 3 defines the binary relationship and its multiplicities. Transformation techniques for relationships with common multiplicity values is presented in section 4. Particular multiplicities and their transformation is explained in section 5. Example of full PIM transformation with relationships with particular multiplicity values is shown in section 6 and final conclusions are given in section 7.

2 Existing tools for model and constraint transformation

There are several tools that provide support for model and OCL constraint evaluation and transformation.

Enterprise Architect [12] is a CASE tools for creating and managing models. It supports model transformation, source code generation and reverse engineering from source code to PSM models. For instance, it includes transformation from platform independent class model to platform specific database model and also generation of SQL source code from the database model. However, these transformations does not consider the minimal multiplicities of a relationship for the required constraints as described further in this paper.

Dresden OCL Toolkit [2,13] is a Eclipse plugin. The toolkit can load a UML class model and OCL constraints. It can load a model instance and evaluate the constraints for it. It can also generate SQL code for a database to create and constraints to check in it or AspecJ source code for constraints checking in Java. However, even this toolkit does not consider the minimal multiplicities of a relationship for the foreign key direction, nullability and uniqueness or specific OCL constraints creation to ensure the multiplicity.

Therefore we want to define the multiplicity constraints in a formal way – in OCL – so they can be adapted in such a tool.

3 Binary relationship and its multiplicities

The conceptual model uses entities or classes as the representation of the types of objects, and associations between them representing relationships between entities. Relationships can be unary (properties of objects), binary (an association between two objects), or n-ary (an association between n objects). It can be proved, that n-ary relationships can be substituted by (n-1) binary relationships without loss of generality [11]. Therefore, we can elaborate only the binary relationships.

Binary relationship is a connection between two entities. It means instances of one entity has a relationship to an instance of the other one (however, it can be the same entity as well).

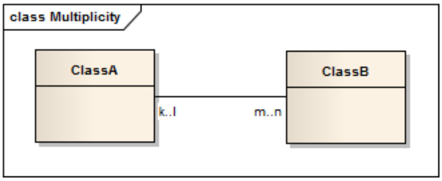


Fig. 1. Labeling of minimal and maximal multiplicities of a relationship in UML class diagram

There are several types of relationship depending on multiplicities of the relationship. The multiplicity defines the number of instances connected to each other. Fig. 1 shows general form of modeling binary relationships using UML class diagram notation [4,5,1]. Multiplicities of the relationship are labeled by parameters k, l, m and n with the following meaning:

- k represents minimal number of instances of ClassA related to an instance of ClassB
- l represents maximal number of instances of ClassA related to an instance of ClassB
- m represents minimal number of instances of ClassB related to an instance of ClassA
- n represents maximal number of instances of ClassB related to an instance of ClassA.

The minimal multiplicity of a relationship indicates the obligation of an instance to be related to instance or instances of the other type. Minimal multiplicity equal to *zero* means that there is no obligation for the source instance to be related to any instance of the target type. On the other hand, when the minimal multiplicity is equal to *one*, it means that the source instance must be related to at least one target instance. The former is actually no restriction at all. The latter is a restriction and it can be expressed using OCL constraint as follows:

```
context a:ClassA inv minBdirect:
not a.b.asSet()->isEmpty()
```

Because the relationship can be unidirectional – and it usually is, for instance in PSM for relational database using foreign keys – it is useful in such case to define the constraint in reverse direction like the following:

```
context a:ClassA inv minBreverse:
ClassB.allInstances()->exists(b | b.a = a).
```

The maximal multiplicity of relationship indicates if the instance can be related just to a single target instance or to a set of target instances. The maximal multiplicity of *one* stands for an instance of source type being related to a single instance of target type, while *asterisk (*)* stands for a set of instances of target type related to a single instance of the source type. The latter one is actually no restriction while the former is a restriction and it can be expressed using OCL constraint as follows:

```
context a:ClassA inv maxBdirect:
a.b.asSet()->size() <= 1
```

or with perspective of unidirectional relationship as follows:

```
context a:ClassA inv maxBreverse:
ClassB.allInstances()->count(b|b.a=a) <= 1
```

However, minimal and maximal multiplicities can take other values. We call such multiplicities particular ones and deal with them in section 5.

4 Transformation of binary relationships to PSM of relational database

In relational databases entities are represented by tables of rows, while rows represent instances of entities [11,8]. Each row can be identified by a mechanism called primary key, which ensures that each record is identified by the nonempty unique key. Relationships between instances of entities are represented by a mechanism of foreign keys, which links a row in the table (representing source entity) to a single row in a table representing target entity – using the target's primary key.

The mechanism of foreign and primary keys brings two main problems. Using this mechanism, any single source row can refer just to a single target row, not a set of rows, and therefore it can only realize one-to-one or one-to-many relationship. This is in contrast to conceptual modeling on PIM level where many-to-many relationships are also possible (and very common). Second problem is that the connection represented by foreign key is unidirectional in contrast to bidirectional relationships used in PIMs. That means in PSM of relational database, source entity has direct link to target entity while target entity does not have direct link back.

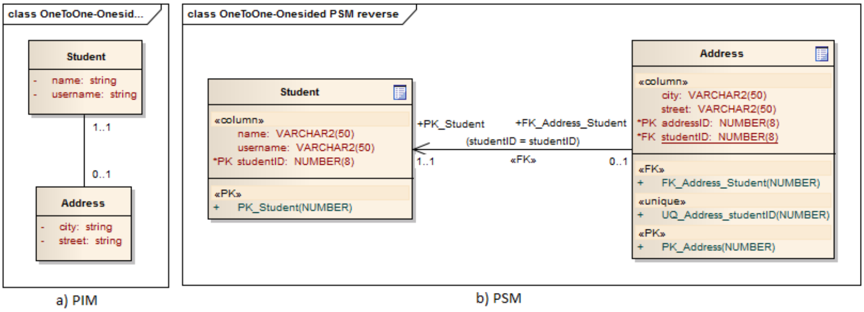


Fig. 2. a) PIM and b) PSM of one-to-one relationship with one side required

However, according to Fig. 2 where the one-to-one relationship is realized by foreign key referring from the table Address to the table Student, this mechanism automatically ensures that the maximal multiplicity of target entity in relationship is always equal to one (l=1). Furthermore, we can determine the minimal target multiplicity to zero (k=0) by making the foreign key *nullable* (i.e. it can be null, referring no target table row) or to one (k=1) by making it *not nullable* (i.e. must not be null, must refer some target table row). Because the foreign key is unidirectional, there is no restriction to the maximal multiplicity of the source entity in relationship and it is implicitly infinite. However, we can restrict it to one (n=1) by making the foreign key *unique*.

There is no way to restrict the minimal multiplicity of the source entity in the relationship (multiplicity m in Fig. 1) using just the foreign key. [11] suggests using *on delete restrict* clause for the foreign key. However, this clause only restricts deletion but not existence of related instance in general. Therefore we suggest defining a special constraint to restrict it to one (m=1) if required. This constraint is defined in sections 4.1 and 4.2. There are also some other restrictions for the realization by the foreign key mechanism depending on the multiplicity of relationship described in sections 4.1, 4.2 and 4.3.

4.1 Transformation of one-to-one relationship

In one-to-one relationship the direction of the foreign key differs depending on the minimal multiplicities of the relationship.

If both minimal multiplicities of the relationship are equal to zero, the direction of the foreign key does not matter in this case.

If the minimal multiplicity of one side of the relationship is equal to one, the direction of the foreign key realization is given as follows: the foreign key must be *not nullable* and must refer from the table representing the not required entity referring to a primary key in the table representing the required entity as shown in Fig. 2b.

If both minimal multiplicities of the relationship are equal to one, the relationship is required by both sides. In this situation there are two possible methods to realize the required relationship in PSM for relational database: using single foreign key; or using pair of reverse foreign keys.

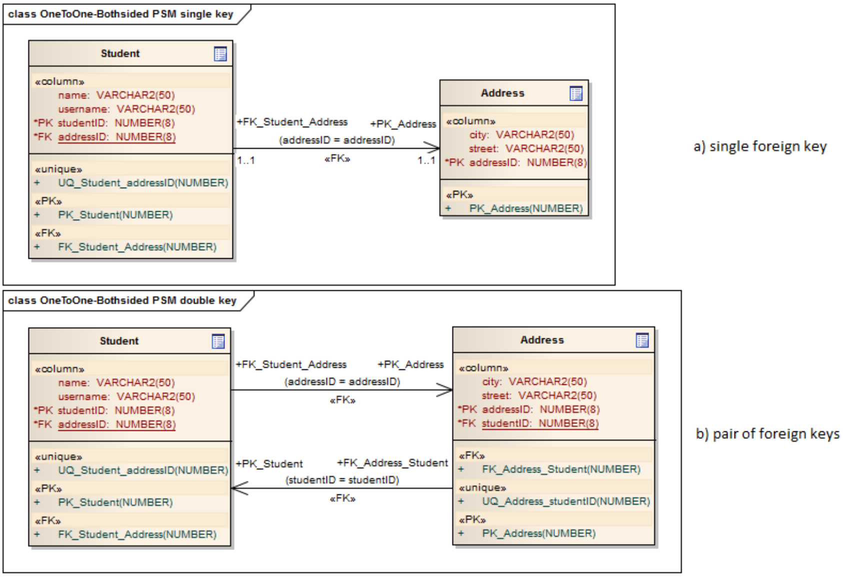


Fig. 3. PSM of one-to-one both-side required relationship with a) single foreign key and b) pair of foreign keys

Using single foreign key. In this case, the direction of the foreign key does not matter again like in the situation where both minimal multiplicities are zero. Let's consider the realization shown in Fig. 3a. To ensure minimal multiplicity k=1 while using only a single foreign key the realization of the reverse definition of the constraint for minimal multiplicity defined in section 3 is necessary to check for existence of the other related instance. The OCL constraint can be realized in SQL as view selecting records violating the multiplicity restrictions. The constraint view for the situation in Fig. 3 can be defined as follows:

```
CREATE VIEW violating_address AS
SELECT a.addressID FROM Address a
WHERE NOT EXISTS (SELECT 1 FROM Student s WHERE s.addressID = a.addressID);
```

This constraint can be also used to realize one-to-one relationship with one side required using the foreign key of reverse direction, i.e. the foreign key referring from the table of the required entity to a primary key in the table of the not required entity. In this case this constraint must be defined to ensure required minimal multiplicity in the opposite direction $k=1$ and the foreign key is *nullable* to enable minimal multiplicity $m=0$.

Using pair of reverse foreign keys. Another possible solution to ensure both minimal multiplicities $k=1$ and $m=1$ in one-to-one both-side required relationship is using pair of reverse *not null unique* foreign keys as shown in Fig. 3b. However, additional constraint must be defined to make sure both foreign keys refer the same instances, i.e. there is no circle of relations between a set of instances of both types like *s1 -> a1 -> s2 -> a2 -> s1*. The constraint can be defined in PIM in OCL as follows:

```
context s:Student inv notCircular: s.add.std = s
```

In PSM for relational database the constraint can be realized as view selecting records violating this constraint as follows:

```
CREATE VIEW violating_circular_students AS
SELECT s.studentID FROM Student s, Address a
WHERE s.addressID = a.addressID and a.studentID <> s.studentID;
```

Single table realization of one-to-one relationship. One-to-one relationship in general can be also realized by a single table representing both related entities. Such table contains all attributes of both entities and each row in such table stands for the whole relationship of instances while in case of no instance related the attributes of not participating instance stay *null*.

However, this realization violates third normal form [11] and the intention of designer to represent the entities separately. If the designer intends to realize the relationship in a single table then he can make this transformation in PIM already.

We prefer the realization using two separate tables and relationship realized by single foreign keys with additional constraints as described above.

4.2 Transformation of one-to-many relationship

One-to-many relationship is realized in PSM for relational database using foreign key like in the realization of one-to-one relationship. In this situation the direction of the key is strictly defined by the maximal multiplicities – the foreign

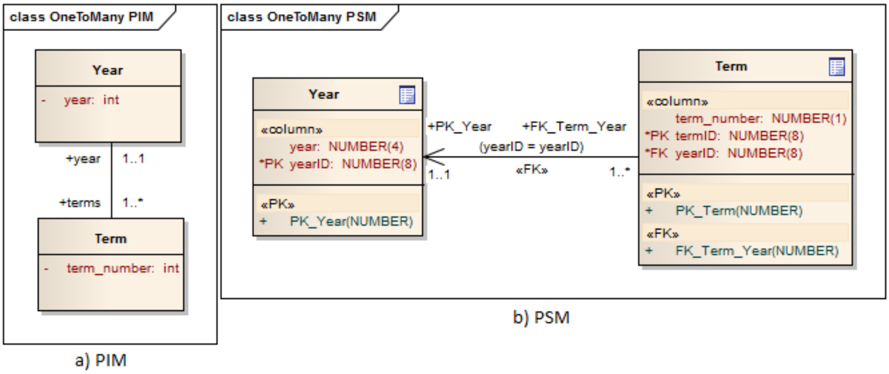


Fig. 4. a) PIM and b) PSM of one-to-many relationship

key must refer from the table representing the entity with higher multiplicity (source) to the table representing the entity with maximal multiplicity of one (target). Transformed PSM for the situation in Fig. 4a is shown in Fig. 4b.

Maximal multiplicity l=1 is ensured by the foreign key mechanism. To enable maximal multiplicity of the other side n=* the foreign key is *not unique*, so many source rows can refer the same target row. If the target entity is required in the relationship (i.e. the minimal multiplicity of target is k=1), the foreign key must be *not null* to ensure the requirement of being related to at least one target record. Finally, if the source entity type is required in the relationship (i.e. the minimal multiplicity of source entity is m=1), additional constraint must be defined and realized as shown in section 4.1.

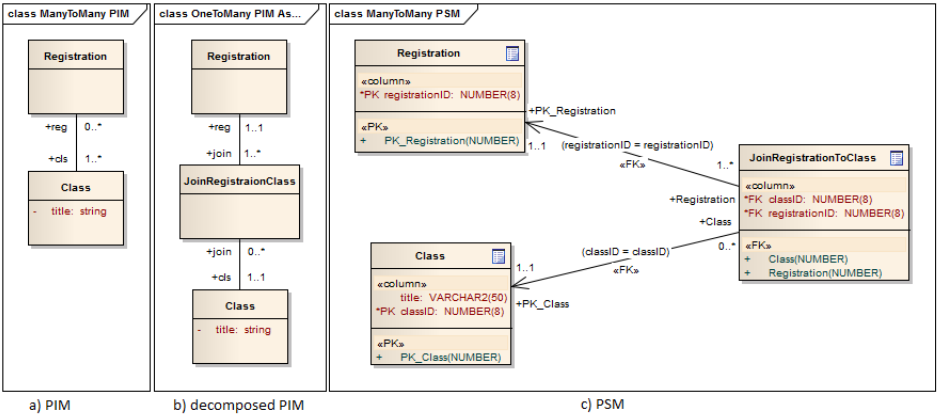
The constraint can be realized in PSM for relational database by the following view selecting records that violate the required multiplicity:

```
CREATE VIEW violating_years AS
SELECT y.yearID FROM Year y
WHERE NOT EXISTS (SELECT 1 FROM Term t WHERE t.yearID = y.yearID);
```

4.3 Transformation of many-to-many relationship

It has been said before that relationships in relational databases are realized by using foreign key which is unidirectional and always refers only to a single row in a table. To realize many-to-many relationship we need to decompose the relationship into two one-to-many relationships and an entity to represent the original relationship (association entity) [11]. Minimal and maximal multiplicities of the original relationship become minimal and maximal multiplicities of the association entity in the appropriate one-to-many relationship. Minimal and maximal multiplicity of the original related entities become one in the relationship to the association entity.

Example of the decomposition of the relationship in PIM level is shown in Fig. 5a and 5b. The many-to-many relationship of entities Class and Registration



a) PIM b) decomposed PIM c) PSM

Fig. 5. PIM of many-to-many relationship

has been decomposed to an entity JoinRegistrationClass representing the relationship between a class and a registration and two one-to-many relationships of Class and JoinRegistrationClass and Registration and JoinRegistrationClass.

After this decomposition the transformation for both one-to-many relationships can be applied as shown in section 4.2. The resulting realization of the many-to-many relationship is shown in Fig. 5c.

5 Binary relationship with particular multiplicity

We have presented methods to transform typical cases of binary relationships from PIM to PSM for relational database. We have defined constraints for typical minimal and maximal multiplicities of relationships and their transformation to SQL views to select rows violating multiplicity constraints.

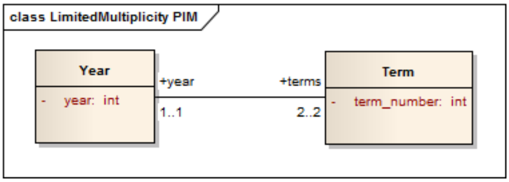


Fig. 6. PIM of one-to-many relationship with particular multiplicity

In many practical modeling situations the multiplicities of relationships in PIM are not restricted to those typical situations mentioned before. The multiplicities can be more restrictive to define exact number of instances being related together. We use the term *particular multiplicity* for minimal multiplicity greater than *one* or for maximal multiplicity other than *** or *1*. An example of such situation of strongly restricted multiplicities is shown in Fig. 6. The figure presents the situation of years and terms again, but this time a year consists of exactly two terms.

The type of relationship and its transformation is still defined by the maximal multiplicities as shown in Section 4. To restrict the particular multiplicity, additional constraints must be defined for minimal and maximal multiplicities. Using the notation shown in Fig. 1 with perspective of unidirectional realization of relationships these constraints can be defined in OCL as follows:

```
context a:ClassA inv minA:
ClassB.allInstances()->count(b | b.a = a) >= m
```

for the minimal multiplicity restriction and

```
context a:ClassA inv minA:
ClassB.allInstances()->count(b | b.a = a) <= n
```

for the maximal multiplicity restriction. If both minimal and maximal multiplicity of one side of a relationship are restricted to particular values, both constraints can be joined together into single constraint:

```
context a:ClassA inv minA:
ClassB.allInstances()->count(b | b.a = a) >= m
&
ClassB.allInstances()->count(b | b.a = a) <= n
```

5.1 Transformation of particular multiplicity constraints

Relationships with particular multiplicities are transformed to PSM for relational database the same way as binary relationships with typical multiplicities as defined and shown in section 4. Additionally, the defined constraints must be realized in SQL to ensure required multiplicity values. This can be done for situation from Fig. 6 by defining views selecting records violating the restricted multiplicities as follows:

```
CREATE VIEW violating_years_minimal AS
SELECT y.yearID FROM Year y
WHERE 2 > (SELECT count(*) FROM Term t WHERE t.yearID = y.yearID);
```

for the minimal multiplicity m=2 and

```
CREATE VIEW violating_years_maximal AS
SELECT y.yearID FROM Year y
WHERE 2 < (SELECT count(*) FROM Term t WHERE t.yearID = y.yearID);
```

for the maximal multiplicity n=2. In this situation, both minimal and maximal multiplicities are restricted to particular values, so a single view can be defined selecting records of years violating the restricted multiplicities generally:

```
CREATE VIEW violating_years AS
SELECT y.yearID FROM Year y
WHERE 2 <> (SELECT count(*) FROM Term t WHERE t.yearID = y.yearID);
```

6 Example transformation

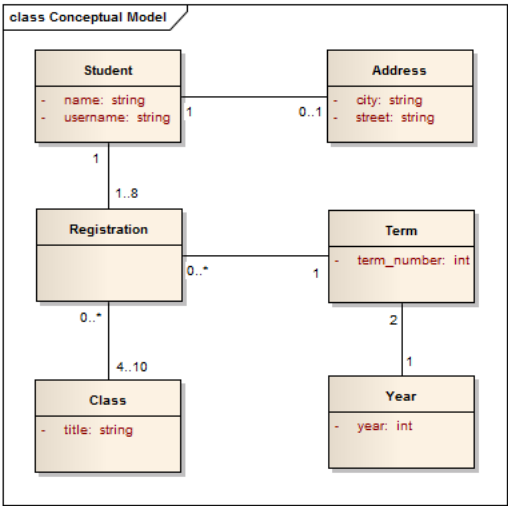


Fig. 7. PIM of the example

To illustrate binary relationships between entities and their transformations to PSM for relational database we prepared an example of a small school system for evidence of students and their registrations of classes to set of terms. Students make a registration of four to ten classes each term they study while each school year consist of exactly two terms. Each student can make eight registrations at most but, on the other hand, it must be registered at least into one term to appear in the system. Students may have address assigned but there might be no addresses without a student assigned. Fig. 7 presents UML class diagram[1] for the PIM of the example.

We chose transformation of PIM to PSM using the single foreign key to represent binary relationships between entities with additional constraints defined in OCL a realized in SQL as views as shown in this paper. The PSM is shown in Fig. 8 and the constraints defined as follows:

```
context s:Student inv countRegistrations:
Registration.allInstances()->count(r | r.std = s) >= 1
&
Registration.allInstances()->count(r | r.std = s) <= 8
```

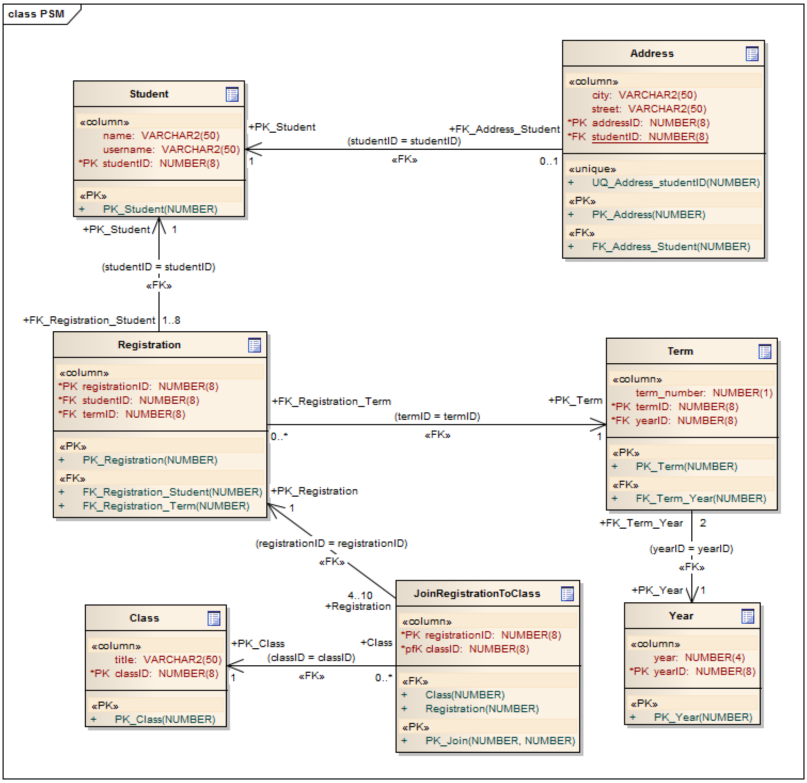


Fig. 8. Transformed PSM of the example

for multiplicity from *1* to *8* of relationship between Student and Registration,

```
context y:Year inv countTerms:
Term.allInstances()->count(t | t.year = y) = 2
```

for multiplicity *2* of relationship between Year and Term, and

```
context r:Registration inv countClasses:
Class.allInstances()->count(c | c.reg = r) >= 4
&
Class.allInstances()->count(c | c.reg = r) <= 10
```

for multiplicity from *4* to *10* of relationship between Registration and Class.

In PSM the defined constraints are realized by views selecting records violating multiplicity restrictions as follows:

```
CREATE VIEW violating_countRegistrations AS
SELECT s.studentID FROM Student s
WHERE (SELECT count(*) FROM Registration r
       WHERE r.studentID = s.studentID) NOT BETWEEN (1,8);
```

for multiplicity from *1* to *8* of relationship between Student and Registration,

```
CREATE VIEW violating_countTerms AS
SELECT y.yearID FROM Year y
WHERE (SELECT count(*) FROM Term t WHERE t.yearID = y.yearID) <> 2;
```

for multiplicity *2* of relationship between Year and Term, and

```
CREATE VIEW violating_countClasses AS
SELECT r.registrationID FROM Registration r
WHERE (SELECT count(*) FROM JoinRegistrationToClasses j
       WHERE j.registrationID = r.registrationID) NOT BETWEEN (4,10);
```

7 Conclusions

In this paper we presented transformations of binary relationships from PIM to PSM for relational databases. We presented basic principles of transformations of relationships according to their multiplicities to relational tables with the mechanism of foreign and primary keys. We also suggested restrictions associated with the relationship realization in PSM to ensure required minimal multiplicities in relationships. We defined these restrictions in PIM using OCL constraints so automated tools can use them for their transformations. We also presented examples of transformation of such constraints to PSM for relational databases as views selecting rows that violate the required multiplicities.

We also defined particular multiplicities of relationships and their expression using OCL constraints. We suggested its transformations to PSM for relational database and showed examples for illustration. We also presented a full example of PIM transformation to PSM for relational database with required constraints as suggested and defined in this paper.

References

1. Arlow, J., Neustadt, I.: UML 2.0 and the Unified Process: Practical Object-Oriented Analysis and Design (2nd Edition). Addison-Wesley Professional (2005)
2. Demuth, B.: DresdenOCL. http://www.reuseware.org/index.php/DresdenOCL (Jan 2011)
3. OMG: Object constraint language, version 1.3. http://www.omg.org/spec/OCL/2.2/PDF (Feb 2010)
4. OMG: Object management group - UML. http://www.uml.org (Feb 2011)
5. OMG: UML 2.3. http://www.omg.org/spec/UML/2.3/ (Feb 2011)
6. OMG, Miller, J., Mukerji, J.: MDA guide version 1.0.1. http://www.omg.org/cgi-bin/doc?omg/03-06-01.pdf (Jun 2003)
7. Richta, K.: Jazyk OCL a modelem řízený vývoj. In: Moderní databáze - Architektura moderních IS 2010 (2010)
8. Richta, K.: Rekonstrukce OCL z SQL. In: Datakon 2010 (2010)
9. Richters, M., Büttner, F., Gutsche, F., Kuhlmann, M.: USE - a UML-based specification environment. http://www.db.informatik.uni-bremen.de/projects/USE/ (Jan 2011)
10. Richters, M., Gogolla, M.: OCL: syntax, semantics, and tools. In: Object Modeling with the OCL, The Rationale behind the Object Constraint Language. pp. 42–68. Springer-Verlag (2002)
11. Rob, P., Coronel, C.: Database Systems: Design, Implementation, and Management. Boyd & Fraser, 2nd edn. (1995)
12. Sparx Systems: Enterprise architect - UML design tools and UML CASE tools for software development. http://www.sparxsystems.com.au/products/ea/index.html (Mar 2011)
13. Wilke, C., Thiele, M., Freitag, B.: Dresden OCL: manual for installation, use and development (Oct 2010)

Towards e-Government Interoperability Framework

Jiri Feuerlicht[1,2], and David Cunek[1]

[1] University of Economics, Prague
nám. W. Churchilla 4 130 67 Praha 3, Czech Republic
[2] University of Technology, Sydney Broadway, NSW, 2007, Australia
jiri.feuerlicht@vse.cz, xcund01@vse.cz

Abstract. Over the past decade most countries with advanced economies have made major investments in various e-Government initiatives attempting to replace traditional paper-based communications between government departments, citizens and private organizations with electronic communications. Notwithstanding such costly initiatives the success of e-Government efforts has been limited. In this paper we discuss e-Government initiatives in the Czech Republic and argue that successful introduction of e-Government must be done in the context of a service-centric interoperability framework that facilitates the development of e-Government services with well-defined and stable interfaces.

Keywords: e-Government, interoperability, data integration, e-Government services, service repositories, SOA

1 Introduction

Over the past decade many countries, including the Czech Republic made major investments in various e-Government initiatives attempting to replace traditional paper-based communications between government departments, citizens and private organizations with modern computer-based systems and electronic communications. While the beginnings of e-Government in the Czech Republic can be traced to the end of the last century, it was only recently that the legislative changes that facilitate e-Government have been introduced and became the subject of public debate. State Information and Communications Policy - e-Czech 2006 [1] that incorporates the concepts the e-Europe 2005 Framework was approved in 2004, and work on a number of initiatives including the Information System of Databoxes (ISDS) [2] and the Government Portal [3] have started shortly afterward.

Currently available public sector services in the Czech Republic can be categorized into three types based on the type of interaction and on the style of interface (Table 1). The type of service consumer (i.e. end user) typically defines the style of the interface; web-based interfaces are widely used for interactions between PA (Public Authority) and citizens, while interactions between PA and business entities typically require a programming interface (i.e. API – Application Programming Interface).

V. Snášel, J. Pokorný, K. Richta (Eds.): Dateso 2011, pp. 39–48, ISBN 978-80-248-2391-1.

Table 1. Categorization of public sector services.

Interaction	Service Type	Service Consumer
PA -> End User	Online information on web	Citizens, Businesses
	Codebooks	Businesses
End User -> PA	Downloadable forms on web	Citizens, Businesses
	Submissions over internet	Businesses
PA <-> End User	Signed email communication	Citizens, Businesses
	Data exchange via Databoxes	Citizens, Businesses
	Marketplace for public contracts	Businesses
	Interoperable services	Businesses

The first type of services involves one way interactions between PA and service consumers. This category of services includes information published on the websites of various government departments such as codebooks for standardized data transfer (e.g. units of measure for the Customs or Data Standard for the Ministry of Health). These kinds of services are widely used and are relatively simple to implement. The second type of services involves citizens or businesses sending data to PA, typically using electronic forms that are downloaded from a website, completed by the end users and uploaded. A more sophisticated version of these types of services are services available via the Government Portal, which offers (in addition to a standard web-based interface) a set of Web Services that can be integrated into client's information system (see section 2.2 for additional discussion). Finally, the third type of services facilitates two-way interactions between PA and service consumers (i.e. citizens or businesses). These interactions can take different forms, for example using secure email with digital signatures or the ISDS [4]. Another form of interaction involves specialized electronic marketplaces for public sector e-Procurement. The most challenging type of two-way interactions involves interoperable services that enable automation of business processes between PAs and business entities. Such services should allow sharing of information among different government departments avoiding duplication and reducing the amount of unnecessary paperwork.

Similar classification of e-Government services is used by the European Interoperability Framework [5] that classifies types of services according to their level of sophistication into four stages: stage 1) publishing information online, stage 2) downloadable forms returned via email or mail, stage 3) online forms submitted electronically, and stage 4) full automation and integration of services. Alternative classifications of e-Government services can be found in the literature, for example in [6].

Implementing interoperable services between government departments and business entities involves overcoming a number of challenges, in particular standardization of data structure and semantics across the public administration domain, and specification of well-designed stable interfaces for various types of e-Government services. Notwithstanding many costly initiatives the success of such e-Government efforts has been limited [7]. There is some evidence that the focus of such initiatives has been mainly on providing technical-level interoperability (i.e. specifying data formats, data exchange and security protocols, etc.) rather than providing an overall interoperability framework for e-Government services.

Furthermore, there is a widespread tendency to use existing (paper-based) documents as the basis for developing electronic documents (i.e. XML data structures) and map the data elements from paper forms directly to corresponding XML message data structures. This is evident in data structures used for various Government Portal services, e.g. Customs Administration, Czech Social Security Administration, etc. This leads to a suboptimal solution with complex message structures that require frequent modifications as new requirements are incorporated into existing applications. This *document-centric* approach attempts to achieve semantic interoperability directly by transformation and mapping of data elements between disparate systems operated by different government agencies (see discussion in section 3).

In this paper we argue that interoperable e-Government services can be best achieved in the context of a service-centric framework that covers the entire services life-cycle and produces well-defined and stable interfaces for e-Government services. To achieve high levels of interoperability, e-Government services must be defined from a global perspective (i.e. not separately for each government department) and stored in a globally accessible service repository. We note that while e-Government services represent a special category of interoperable services, experience with the design of services in other domains, e.g. travel [8] can provide useful guidelines for developing a suitable methodological framework.

In the following section (section 2) we discuss e-Government initiatives in the Czech Republic with specific reference to the European Interoperability Framework (EIF). The next section (section 3) describes the document-centric and service-centric interoperability models, and in the final section (section 4) we discuss the limitations of the traditional approach to e-Government interoperability and advocate a service-based interoperability framework.

2 e-Government Interoperability Framework

Interoperability of e-Government applications has been the subject of extensive investigations recently [9-12]. There is a wide agreement that interoperability cannot achieved purely on the basis of providing connectivity at the technical level and that effective sharing of information among government departments requires the support of a methodological framework.

The European Interoperability Framework (EIF) [13] represents an attempt to provide an architectural solution to facilitate automation of business processes among public administrations, citizens and businesses within the European Union. European Interoperability Framework defines four interoperability dimensions: legal, organizational, semantic, and technical that operate within a political context. Political context is set by government strategies; two documents define strategy for e-Government in the Czech Republic. “National Information Strategy” [14] published in 1999 that proposes the creation of *information society* and advocates the integration of information systems in public administration, and “National Information and Communication Strategy” (e-Czech) [15]. This strategy incorporates the concepts of e-Europe 2005 [16], and describes the development of modern online services for the public sector (i.e. e-Government, e-Learning, e-Health) and the creation of a dynamic environment for e-Business. The most recent document setting out strategy for the development of digital economy "A European Information Society for growth and employment" (i2010) [17] represents a continuation of efforts to build information society based on electronic services and the emerging services economy through incorporation of the European Interoperability Framework into the national framework.

Ensuring legal interoperability typically involves introducing new legislations, for example the introduction of Information System of Data Boxes necessitated approval of Act 300/2008 (Digital agenda), and the Government Portal is supported by Act 365/2000 (Information systems in public administration). Other important legislations in this area include Act 111/2009 (Basic registers) and Act 227/2000 (Digital signature).

Another aspect of e-Government interoperability that needs to be addressed is alignment of organizational structures and processes in different government departments (Organizational Interoperability). According to the EIF requirements of service consumers (i.e. requirements of businesses and citizens) need to be taken into account to minimize duplication when interacting with various government departments. Consider, for example a situation where an individual is setting up a new company and needs to provide various documents including criminal record, information about social security debts, etc. Such information is already available in various government departments and should be accessed automatically. This in practice implies business process re-engineering and alignment so that individual government departments share information, avoiding the need for the clients to provide information that is already held by different government departments. The pre-requisite for effective data sharing between government departments is semantic interoperability, i.e. the standardization of data structures and semantics of data elements used across government agencies.

2.1 Semantic-Level Interoperability

Semantic interoperability deals with the meaning of data exchanged among the various agencies in public administration and requires that data heterogeneity present in disparate information systems operated by individual government departments is resolved. To address semantic interoperability EU established the SEMIC.EU (Semantic Interoperability Centre) for member states to exchange information about the meaning of data elements, which following approval by all participants are published in a data elements repository.

At a national level, in the Czech Republic, metadata about information exchanged between various PA departments is maintained using Information System about Data Elements (ISDP) [18]. Public agencies are obligated to use data elements published in the ISDP database when communicating electronically between PA departments. However, ISDP data elements are not required for the implementation of interfaces with private sector companies. The Czech Ministry of Interior has published a document "Methodological Guidance for Creation of Data Elements" that describes data element semantics and includes recommendations about data element life cycle management. Another related document "Method of development of XSD schemas in public administration" deals with technical issues that concern the development of XML schemas and describes basic data types, composition of basic data types into complex types and conventions for the design of schemas (e.g. namespaces, naming of elements, etc.). ISDP maintains metadata information about data elements used in public administration, but it should be noted that it is not a repository of definitions of service interfaces. Although the use of ISDP data elements is mandated for public agencies, the definition of interfaces for specific services is the responsibility of individual service providers, typically government departments. This approach is consistent with EIF, but it limits interoperability as services implemented by a given government department will be, in general incompatible with services developed by other departments, although their functionality may be identical. Another important consequence of this approach is that there is no centralized repository of e-Government services, potentially causing duplication and inconsistencies in service definitions across various agencies of public administration.

2.2 Technical-Level Interoperability

Technical interoperability deals with technical aspects of interconnecting PA systems, in particular with standardization of data formats and data exchange protocols (e.g. SOAP, AS2). Other technical issues include ensuring performance, scalability and security (identification, authorization, integrity, non-repudiation, encryption, time-stamping, and protection against attacks and viruses). When developing interoperable services, such issues have to be considered and resolved to provide a technical solution. Two distinct communication channels are available for e-Government applications to enable technical-level interoperability among PA agencies, citizens, and private organizations: the Government Portal and ISDS.

The transactional component of the Government Portal is a gateway that enables interactions between citizens or businesses (using Government Portal standards and protocols) with PA institutions' Department Interface Servers (DIS). Government Portal implementation platforms is MicroSoft Biztalk [19] that manages data flows between parties (including schema transformation, message routing, etc.) and can be used to implement business processes that span government departments. Secure communication between parties is achieved using transmission of XML messages over HTTPS protocol. Messages are specified using XSD schema and consist of an envelope containing routing information for a specific DIS server and data payload. While the technical aspects of the Government Portal communications are documented on the portal (Operating Rules for Community Portal for Developers), as already noted, message specifications used by individual government departments are not centrally managed.

While the Government Portal provides a standard gateway for the implementation of e-Government services, ISDS emulates an email solution. Information System of ISDS provides user interface similar to standard email and enables sending and receiving of data messages that are stored in Databoxes. All Databoxes are stored in one central location, so that sending and receiving messages does not involve data transmission, but is implemented by linking sender and receiver records. In addition to a web-based user interface, ISDS provides an API that enables connectivity via SOAP and HTTPS. Using this public interface, users can integrate ISDS functionality into their applications. The API is implemented as a set of Web Services (e.g. CreateMessage, MessageDowload, etc.), that operate on data messages specified using XSDs and consists of an envelope containing a signature and a timestamp and data payload. In line with the ISDP approach, ISDS specification does not define the structure of the data payloads, making the automation of messages processing impossible.

3. Information-level Interoperability models

To fully appreciate the extent of the interoperability challenge and to explore alternative solutions we need to refine the interoperability framework introduced in the previous section (section 2). Interoperability has been the subject of extensive investigation in the context of B2B (Business to Business) applications [20], [21] and has been classified into three different types (levels): technical, information, and business process interoperability [22]. Technical-level interoperability deals with disparate communication protocols, language environments, and technology platforms that are used by partner organizations and directly corresponds to technical-level interoperability in the EIF framework described in section 2.2. Information-level interoperability concerns data heterogeneity, and can be further classified into: syntax, structure, and semantic heterogeneity. Syntax heterogeneity refers to differences in formats used to represent data (e.g. XML, tagged document formats, etc.). Individual organizations often use schemas with different structure of business documents to represent the same information. Such schema differences are referred to as structure heterogeneity. Semantic heterogeneity concerns the differences in the meaning of individual data items and business process-level interoperability is concerned with collaborative activities between the partners.

While technical-level interoperability is today largely resolved by adopting appropriate standards (i.e. XML, HTTP, SOAP, etc.) as illustrated in section 2, information-level and business process-level interoperability issues remain a significant challenge, in particular in environments that involve a large number of autonomous partner organizations. Discussion of business process-level interoperability is outside the scope of this paper. Focusing on information-level interoperability, two basic models have been documented in the literature: document-centric and service-centric models [23].

3.1 Document-Centric Interoperability Model

Most existing implementations (including the e-Government interoperability framework described in section 2) adopt the document-centric interoperability model characterized by shipping business documents between partner systems. The main advantage of this approach is that the use of documents as basic artifacts of business communication avoids the need for compatibility of underlying technology platforms, so that technical-level interoperability can be relatively easily achieved. Business documents provide a level of abstraction that allows automation of inter-enterprise business processes based on a mutual understanding of the structure and semantics of documents. However, document-centric approach suffers from limited scalability associated with its reliance on document translation, i.e. as the number of partners and the complexity of their data structures increases, the mappings between individual schema elements becomes unmanageable.

Another principal limitation of the document-centric approach is its tendency to use large and complex message structures that typically mirror the original paper-based forms. The complexity of message structures arises from designing message payloads to include all the information needed to perform the corresponding business function without any reference to information received in previous messages, making the interaction essentially stateless. While such stateless interactions reduce the number of messages needed to implement a particular business function, it results in complex and redundant message data structures making changes to message formats difficult to perform without introducing undesirable side-effects that invalidate existing applications. The message payloads form the interface between applications and therefore introduce high levels of data coupling and interdependencies by externalizing complex document data structures [24]. In effect, the document-centric approach attempts to solve a *data integration problem* for a potentially large number of participants with diverse schemas and business semantics [25], and as is evident from decades of research in the area of integration of heterogeneous databases, this problem does not have a satisfactory solution.

3.2 Service-Centric Interoperability Model

The emergence of SOA (Service-Oriented Architecture) and Web Services provides an opportunity for a new approach to addressing the interoperability challenge. Web Services and SOAP (or REST services [26]) remove the need to use document interchange as an interoperability mechanism by providing technical-level interoperability. Service-centric interoperability model relies on well-defined service interfaces that typically implement a single business function as a Web Service operation. This reduces the problem of standardizing document formats and data semantics to a more manageable task of standardizing service interfaces for a given application domain. Such domain-specific service interfaces are conceptually similar to APIs that are used extensively in programming environments. Interoperability of service-oriented applications relies on stable service interfaces used consistently across the e-Government application domain. Standardization of services ensures that service providers (i.e. different government agencies and departments) publish identical interfaces, avoiding the need to interpret interface semantics. The key difference between the document-centric and the service-centric interoperability models is that service interfaces can be designed to minimize interdependencies (i.e. coupling) between services by encapsulating message data structures and exposing method (operation) signatures that constitute a stable contract between the service provider and the service consumer. Data engineering principles can be applied to the design of service interfaces maximizing service cohesion and resulting in stable and maintainable services [27], [28].

4. Conclusions

It is evident from our discussion in section 2 that the e-Government interoperability framework adopted in the Czech Republic and based on the European Interoperability Framework provides effective solution for technical-level interoperability. However, a number of issues remain to be resolved to achieve information-level interoperability. While the ISDP system maintains metadata information about data elements, and the use of these standard elements is mandated for public agencies, the use of ISDP data elements is not prescribed for private sector companies. Furthermore, the definition of interfaces for specific services remains the responsibility of individual service providers, so that while semantic consistency at the data element level is assured, it is highly likely that individual government agencies will produce incompatible services with overlapping functionality. The absence of a centralized repository of e-Government services makes it difficult to avoid duplication and inconsistencies in service definitions.

With growing acceptance of service-oriented computing and successful application of services in application domains such travel [29] and healthcare [30], it is likely that a service-centric interoperability framework would bring similar benefits to e-Government applications. An important advantage of the service-centric approach is that service interfaces can be designed to significantly limit the exposure of metadata, considerably improving the stability and robustness of e-Government applications. Other advantages of the service-centric approach include improved software reliability simplified development, and support for evolution of interfaces [31].

A key requirement for achieving semantic interoperability in service-oriented applications is standardization of service interfaces for a given application domain (i.e. e-Government). Without such interface standards, equivalent services published by different providers will not be compatible, placing a burden for resolving the inconsistencies on service consumers.

Semantic interoperability is a necessary, but not sufficient condition for the automation of e-Government interactions. Equally, adopting the service-centric approach does not eliminate the need to agree on information and business process semantics, but it makes the problem more manageable by limiting the scope of agreement to service interfaces that typically correspond to simple operations (e.g. change of residence). Well-designed service interfaces address the equally important problem of limiting exposure of complex and often redundant data structures typical of the document-centric approach. Moving away from an interoperability model based on document interchange and adopting a service-centric approach results in a higher level of abstraction associated with the use of application programming interfaces. As experience with programming APIs demonstrates, the benefits of standardized interfaces include improved software reliability, reusability, extensibility, and maintainability, and can ultimately lead to significant application development productivity gains.

Finally, it is evident that in order to address information-level and business process interoperability issues data and business processes that are currently *owned* by different departments have to re-engineered and integrated into a consistent set of e-Government services maintained in a centralized service repository.

Acknowledgments

This research was supported by GACR (Grant Agency, Czech Republic) grant No. GA406011 P403/11/0574.

References

1. *Státní informační a komunikační politika*. 2006 [cited 02-02-2011]; Available from: http://knihovnam.nkp.cz/docs/SIKP_def.pdf.
2. *Informační systém datových schránek (ISDS)*. 2010 [cited 02-02-2011]; Available from: http://www.mvcr.cz/clanek/informacni-system-datovych-schranek-isds.aspx.
3. *Portal verejne spravy*. 2011 [cited 02-02-2011]; Available from: http://portal.gov.cz/wps/portal/_s.155/6966/place.
4. *Information System of Databoxes (ISDS)*. [cited 02-02-2011]; Available from: http://www.datoveschranky.info/.
5. Chen, D. and G. Doumeingts, *European initiatives to develop interoperability of enterprise applications--basic concepts, framework and roadmap*. Annual Reviews in Control, 2003. **27**(2): p. 153-162.
6. Curtin, G.G., *Issues and Challenges: Global EGovernment/E-Participation, Models, Measurement and Methodology*. E-Participation and EGovernment: Understanding the Present and Creating the Future, 2006.
7. *Otesanek Egon*. 2010 [cited]; Available from: http://ekonom.ihned.cz/c1-44167410-otesanek-egon.
8. OTA. *OTA Specifications*. 2010 [cited 6 May 2010]; Available from: http://www.opentravel.org/Specifications/Default.aspx.
9. Otjacques, B., P. Hitzelberger, and F. Feltz, *Interoperability of e-government information systems: Issues of identification and data sharing*. Journal of Management Information Systems, 2007. **23**(4): p. 29-51.
10. Gottschalk, P. and H. Solli-Saether, *Stages of e-government interoperability*. Electronic Government, an International Journal, 2008. **5**(3): p. 310-320.
11. Gottschalk, P., *Maturity levels for interoperability in digital government*. Government Information Quarterly, 2009. **26**(1): p. 75-81.
12. Scholl, H. *Interoperability in e-Government: More than just smart middleware*. 2005: IEEE. 0769522688.
13. *European Interoperability Framework (EIF) Version 2*. 2004 [cited 02-02-2011]; Available from: http://ec.europa.eu/isa/strategy/doc/annex_ii_eif_en.pdf.
14. *National Information Strategy*. 1999 [cited 02-02-2011]; Available from: http://www.epractice.eu/files/media/media_425.pdf.
15. *State Information and Communications Policy (e-Czech 2006)*. 2006 [cited 04-02-2011]; Available from: http://www.epractice.eu/node/281047.
16. *eEurope 2005*. 2005 [cited 07-02-2011]; Available from: http://europa.eu/legislation_summaries/information_society/l24226_en.htm.
17. *i2010 - A European Information Society for growth and employment*. 2010 [cited 06-02-2011]; Available from: http://ec.europa.eu/information_society/eeurope/i2010/index_en.htm.
18. *INFORMAČNÍ SYSTÉM O DATOVÝCH PRVCÍCH* 2010 [cited]; Available from: https://www.sluzby-isvs.cz/ISDP/DefaultSSL.aspx.
19. Microsoft. *Microsoft BizTalk Server: Home*. 2010 [cited 02-02-2011]; Available from: http://www.microsoft.com/biztalk/default.mspx.

20. Ouksel, A., Sheth, A. , *Semantic Interoperability in Global Information Systems: A brief introduction to the research area and the special section.* SIGMOD Record Special Interest Group on Management of Data, 1999. **28**(1): p. 5-12.

21. Medjahed, B., et al., *Business-to-business interactions: issues and enabling technologies.* The VLDB Journal, 2003. **12**(1): p. 59-85.

22. Bussler, C., *B2B Protocol Standards and their Role in Semantic B2B Integration Engines.* Bulletin of the Technical Committee on Data Engineering, 2001. **24**(1): p. 3-11.

23. Feuerlicht, G., Meesathit, S.,. *Service-Centric Interoperability Model for Inter-Enterprise Applications*. in *2nd InterOP Workshop, in conjunction with the 9th IEEE International EDOC Conference*. 2005. Enschede, The Netherlands. 952-10-2698-7.

24. Feuerlicht, G., Wijayaweera, A. . *Determinants of Service Resuability*. in *6th International Conference on Software Methodologies, Tools and Techniques, SoMet 06*. 2007. Rome, Italy: IOS Press. 0922-6389.

25. Nagarajan, M., et al., *Semantic interoperability of web services-challenges and experiences*. 2006.

26. Fielding, R.T. *Architectural Styles and the Design of Network-based Software Architectures*. 2000 [cited]; Available from: http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm.

27. Feuerlicht, G., *Design of service interfaces for e-business applications using data normalization techniques.* Information Systems and E-Business Management, 2005. **3**(4): p. 363-376.

28. Feuerlicht, G., Meesathit, S. *Design framework for interoperable service interfaces*. in *2nd International Conference on Service Oriented Computing*. 2004. New York, NY, USA: ACM Press. 1-58113-871-7.

29. Papazoglou, M., B. Krämer, and J. Yang. *Leveraging Web-services and peer-to-peer networks*. 2010: Springer.

30. Ko, L.F., et al. *HL7 middleware framework for healthcare information system*. 2006: IEEE. 0780397045.

31. Bieber, G. and J. Carpenter. *Introduction to Service-Oriented Programming (Rev 2.1)*. 2001 (updated 9-Apr-01) [cited 27 June]; Available from: http://www.openwings.org/download/specs/ServiceOrientedIntroduction.pdf.

XML Document Versioning and Revalidation*

Jakub Malý, Jakub Klímek, Irena Mlýnková, and Martin Nečaský

XML Research Group, Department of Software Engineering
Faculty of Mathematics and Physics, Charles University in Prague
Malostranské náměstí 25, 118 00 Praha 1
The Czech Republic
{klimek,maly,mlynkova,necasky}@ksi.mff.cuni.cz

Abstract. One of the prominent characteristics of XML applications is their dynamic nature. When a system grows and evolves, old user requirements change and/or new requirements accumulate. Apart from changes in the interface, it is also necessary to modify the existing documents with each new version, so they are valid against the new specification. The approach presented in this work extends an existing conceptual model with the support for multiple versions of the model. Thanks to this extension, it is possible to define a set of changes between two versions of a schema. This work contains an outline of an algorithm that compares two versions of a schema and produces a revalidation script in XSL.

Keywords: XML, schema, schema evolution

1 Introduction

Recently, XML [12] has become a corner stone of many information systems. It is a de facto standard for data exchange (i.e. Web services [4]) and it is also a popular data model in databases [3]. XML schemas are utilized in two scenarios: 1) to describe structure and check validity of internal documents, and 2) to define the interface of a component of the system for other components or of the system to the outer world.

Requirements change during the life cycle of the system and so do the XML schemas. Without any tools to help, the old and new schema need to be examined by a domain expert. Each change must be identified, analyzed and all the relevant parts of the system modified and the existing documents updated. This can be a time-consuming and error-prone process, but, in fact, a significant portion of the operations could be performed automatically.

The existing approaches work directly at the level of XML schemas, which leads to problems with recognizing the semantics of each change. Our approach utilizes the conceptual model for understanding the semantics. Also, most of the existing approaches require the user to either revalidate the documents after

* This work was supported in part by the Czech Science Foundation (GAČR), grant number P202/10/0573, and by the grant SVV-2011-263312.

every performed evolution operation, other require at least all the operations to be performed in one session and directly modify the old version (effectively replacing it with the new version). Our approach imposes no limits on the amount of operations performed between the versions and the amount of versions present in the systems.

The main aim of this paper is to simplify the whole process of system evolution and make the transition to the new version semi-automatic using our conceptual XML modeling framework with the support for multiple versions. We carry on by formally defining possible changes between versions. When changes are detected, we use this information to decide, whether the revalidation is necessary and if it is the case, we generate a script that revalidates existing XML documents against the new version.

1.1 Outline

The rest of this paper is organized as follows: in Section 2, we analyze the existing approaches. In Section 3, we introduce the XSEM conceptual model. In Section 4, we extend the model with versioning support. In Section 5, we formally define changes between versions. In Section 6, we outline how a revalidation script is generated. Section 7 concludes and lists the open problems and areas of future research.

2 Related Work

Approaches can be categorized on the basis of the level where the changes are made by the designer and detected by the algorithm as suggested in [10]:

- in XML schemas (in one of the XML schema languages), so-called *logical* level. At the logical level, changes are detected directly in the evolved schema.
- in diagrams visualizing the XML schema (diagrams directly visualizing constructs of a specific XML schema language).
- in a UML diagram used to model an XML format (usually annotated using comments or stereotypes to specify the translation to an XML schema).

X-Evolution, proposed in [5], is an example of a system built upon a graphical editor for creating schemas in the XML Schema language [14]. At first, the authors defined a set of *evolution primitives*, i.e. basic operations that modify the evolved schema (e.g. *insert_local_element*, *remove_type*). Impact of each primitive on the validity of the existing XML documents is examined. The presented *Adapt* algorithm takes as an input an evolution primitive, an XML schema and an XML document valid against the schema and returns an evolved document valid against the schema on which the evolution primitive is applied.

X-Evolution system provides the user with a narrow set of evolution primitives that can be performed upon XML schemas of a certain format. There are no links to the conceptual model which could be utilized when new content needs to be added during document revalidation; thus, it is up to the user to write the adaptation clauses or update the documents after revalidation. Some very common real-world evolution operations are not considered in the evolution primitives, e.g. moving content, adding a wrapping element for elements or transforming attributes to subelements and vice versa. All algorithms and operations expect a single evolution primitive as an input. And only a single change is supported in each evolution cycle, evolution with multiple changes is not elaborated.

XML Evolution Management, XEM [6] is an approach to manage schema evolution where DTD is used as a schema language. It deals both with changes in DTD and XML documents (*instance documents*).

The proposed primitives are divided into two categories: 1) changes in DTD 2) changes in instance documents. It is proven that the proposed set of primitives is complete, in the sense that any possible change in DTD can be expressed as a sequence of application of the primitives. However, the method used in the proof in many common cases degenerates in removing a large part of the tree and creating it from scratch (e.g. when an element is renamed, it must be removed and then added under new name; if the root element is removed, the whole XML document is first deleted and its structure recreated again). In this process, the structure is created properly by the algorithm, but the data is lost. Again, as in the case of *X-Evolution*, the conceptual model is not considered and thus cannot be utilized when new content is added in the instance documents.

Unlike the previous approaches, *CoDEX*, Conceptual Design and Evolution of XML schemas [7], allows the user to perform a sequence of evolution operation before the documents are revalidated. Each user's single action is recorded by the logging component. When the user is satisfied with the new version of the model, the recorded design steps are minimized and normalized. The proposed conceptual model is closer to a visualization of the XML Schema language than to a platform-independent model and the lack of a conceptual model is directly responsible for problems declared as in-built (e.g. the algorithm can not detect such a change where the same structural element acquires different semantics).

3 XSEM Conceptual Model

Conceptual modeling is a top-down approach to system and data design, which concentrates on correct depicting the problem domain – defining the particular concepts comprising the system and relationships between these concepts.

XSEM is an approach to modeling XML data using *Model Driven Architecture* (MDA) [8]. A prototype implementation of XSEM in a CASE[1] tool called XCase [1] is available. The model has two interconnected layers – platform-independent model (PIM) which utilizes UML [11] class diagrams and platform-specific model (PSM) which again utilizes UML class diagrams but extended with so-called *XSEM profile*. In the rest of this paper, we will use the formal definitions of PIM (Definition 1) and PSM (Definition 2) levels.

[1] Computer-aided software engineering

3.1 PIM Level

A schema in a platform-independent model (PIM) models real-world concepts and relationships between them without any details of their representation in a specific data model (XML in our case). As a PIM, we use the classical model of UML class diagrams in our work. This model is widely supported by the majority of tools for data engineering and there exists the XMI [2] standard for exchanging diagrams between them. Therefore, it is natural to use UML in our approach as well. We introduce PIM schemas formally in Definition 1.

Definition 1. *A* platform-independent schema (PIM schema) *is a triple* $\mathcal{S} = (\mathcal{S}_c, \mathcal{S}_a, \mathcal{S}_r)$ *of disjoint sets of* classes, attributes, *and* associations, *respectively. In addition, PIM schema defines functions* name, card, type, class *and* participant*:*

- Class $C \in \mathcal{S}_c$ *has a name* $name(C)$.
- Attribute $A \in \mathcal{S}_a$ *has a name, data type and cardinality* $name(A)$, $type(A)$ *and* $card(A)$, *respectively.* A *is associated with a class from* $\mathcal{S}_c$ $class(A)$.
- Association $R \in \mathcal{S}_r$ *is a set* $R = \{E_1, E_2\}$ *where* E_1 *and* E_2 *are called* association ends *of* R. R *has a name* $name(R)$. $E \in R$ *has a cardinality* $card(E)$ *and is associated with a class from* $\mathcal{S}_c$ $participant(E)$. *For* R, $participant(E_1)$ *and* $participant(E_2)$ *are called* participants *of* R.

For simplicity, we will also use the notation $attributes(C)$ *to denote the set* $\{A \in \mathcal{S}_a : class(A) = C\}$ *and* $associations(C)$ *to denote the set* $\{R \in \mathcal{S}_r : (\exists E \in R)(participant(E) = C)\}$ *for each class* $C \in \mathcal{S}_c$.

PIM schema components have a usual semantics. A class models a real-world concept. An attribute of that class models a property of the concept. And, an association models a kind of relationships between two concepts modeled by the connected classes. We display each PIM schema as a UML class diagram.

Example 1. Figure 1 shows a sample PIM in XSEM modeling a component accepting purchases. Classes (`Address`, `Purchase`, ...) are depicted as boxes with the list of attributes (`street`, `city`, ...), associations as labeled lines connecting classes (`bill-to`, `ordered`, ...).

3.2 PSM Level

A schema in a platform-specific model (PSM) models how a part of the reality modeled by the PIM schema is represented in a particular data model. In our case, we consider XML as the data model and our PSM schema specifies representation in a particular XML format. Our PSM allows for specifying more different XML formats on the base of the same PIM schema. As a PSM we also use UML class diagrams extended for the purposes of XML data modeling. We introduce PSM formally in Definition 2.

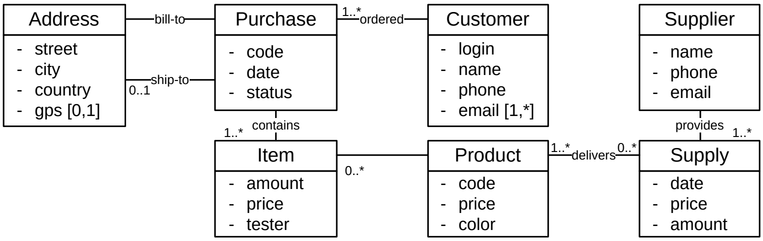


Fig. 1. Sample PIM schema

Definition 2. *A* platform-specific schema (PSM schema) *is a 5-tuple* $\mathcal{S}' = (\mathcal{S}'_c, \mathcal{S}'_a, \mathcal{S}'_r, \mathcal{S}'_m, \mathcal{C}'_{\mathcal{S}'})$ *of disjoint sets of* classes, attributes, associations, *and* content models, *respectively, and one specific class* $\mathcal{C}'_{\mathcal{S}'} \in \mathcal{S}'_c$ *called* schema class. $\mathcal{S}'$ *must be a forest with one of its trees rooted in* $\mathcal{C}'_{\mathcal{S}'}$. *In addition, PSM schema defines functions* name', card', type', xform', cmtype' class' *and* participant'*:*

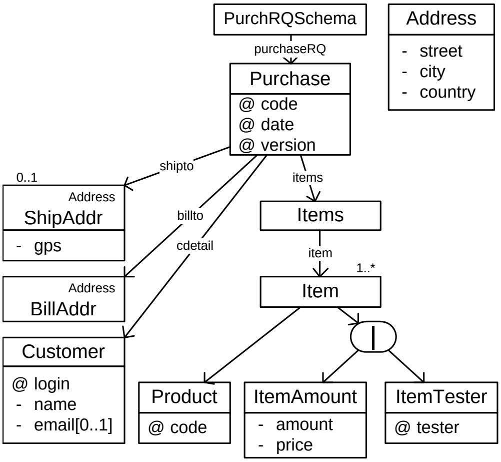
- Class $C' \in \mathcal{S}'_c$ *has a name denoted* $\mathit{name}'(C')$.
- Attribute $A' \in \mathcal{S}'_a$ *has a name, data type, cardinality and XML form denoted* $\mathit{name}'(A')$, $\mathit{type}'(A')$, $\mathit{card}'(A')$ *and* $\mathit{xform}'(A') \in \{e, a\}$, *respectively. Moreover, it is associated with a class from* $\mathcal{S}'_c$ *denoted* $\mathit{class}'(A')$ *and has a position denoted* $\mathit{indexOf}(A')$ *within the all attributes associated with* $\mathit{class}'(A')$.
- Association $R' \in \mathcal{S}'_r$ *is a pair* $R' = (E'_1, E'_2)$ *where* E'_1 *and* E'_2 *are called* association ends *of* R'. $E' \in R'$ *has a cardinality denoted* $\mathit{card}'(E')$ *and is associated with a class from* $\mathcal{S}'_c$ *denoted* $\mathit{participant}'(E')$. $\mathit{participant}'(E'_1)$ *and* $\mathit{participant}'(E'_2)$ *are called* parent *and* child *of* R *and denoted* $\mathit{parent}'(R')$ *and* $\mathit{child}'(R')$, *respectively.* R' *has a name denoted* $\mathit{name}'(R')$ *and has a position denoted* $\mathit{indexOf}(R')$ *among associations with the same parent.*
- Content model $M' \in \mathcal{S}'_m$ *has a content model type* $\mathit{cmtype}'(M') \in \{$`sequence`, `choice`, `set`$\}$.

For simplicity, we will also use the notation $\mathit{attributes}(C')$ *to denote the sequence* $(A'_i \in \mathcal{S}'_a : \mathit{class}(A') = C' \wedge i = \mathit{indexOf}(A'))$ *for each class* $C \in \mathcal{S}_c$.

Let us denote $\mathcal{N}' = \mathcal{S}'_c \cup \mathcal{S}'_m$. *The graph* $\widehat{\mathcal{S}'} = (\mathcal{N}', \mathcal{S}'_r)$ *with classes and content models as nodes and associations as ordered edges must be a directed forest with one of its trees rooted in the schema class* $C'_{\mathcal{S}'}$. *Members of* $\mathcal{S}'_c$, $\mathcal{S}'_a$, $\mathcal{S}'_r$, *and* $\mathcal{S}'_m$ *are called* components *of* $\mathcal{S}'$. *The set of all PSM schemas will be denoted* $\mathcal{S}^{*\prime}$. *Finally we define the set of all constructs (PIM and PSM):* $\mathcal{M} = \mathcal{S}_c \cup \mathcal{S}_r \cup \mathcal{S}_a \cup (\bigcup_{\mathcal{S}' \in \mathcal{S}^{*\prime}} (\mathcal{S}'_c \cup \mathcal{S}'_r \cup \mathcal{S}'_a))$.

The PSM constructs in a concrete PSM schema are linked to the PIM schema constructs (we will say that the PSM schema is derived from the PIM schema and the PSM constructs have an interpretation in the PIM). We will not include formal definition of interpretation in this paper and use these terms intuitively instead.

Example 2. Figure 2(a) shows a sample PSM schema modeling an XML document with a purchase. Again, classes, associations and attributes are depicted



(a) PSM schema

```
<purchaseRQ code="#"
  date="#" version="#">
  <shipto gps="#">
    <street>#</street>
    <city>#</city>
    <country>#</country>
  </shipto>
  <billto>
    <street>#</street>
    <city>#</city>
    <country>#</country>
  </billto>
  <cdetail login="#">
    <name>#</name>
  </cdetail>
  <items>
    <item code="#" tester="#"/>
    <item code="#">
      <amount>#</amount>
      <price>#</price>
    </item>
  </items>
</purchaseRQ>
```

(b) XML document

Fig. 2. Sample PSM schema of a purchase order request with a sample XML document

similarly as in PIM schemas, the difference is in associations, which are now directed. The diagram also contains one content model of type `choice` (parent node of classes `ItemAmount` and `ItemTester`). The XML document depicted in Figure 2(b) illustrates how PSM schema models XML documents. Briefly, named associations model element-subelement nesting, unnamed association inline the content to the element above (see association `Item-Product`), associations starting at the schema class model allowed root elements. Attributes model XML attributes or elements with simple content (depending on the value of *xform'* for the attribute, value `a` is depicted by `@` before the name of the attribute). Structural representatives are used for content reuse (see `ShipAddr` and `BillAddr` reusing content of `Address`). The in-depth description of the relation of PSM schemas and modeled documents can be found in [9].

4 Evolution

For the goal of determining whether XML documents were invalidated due to schema evolution, the system must recognize and analyze the differences between the old ($\mathcal{S}'$) and new ($\widetilde{\mathcal{S}'}$) version of the schema. There are two possible approaches: 1) recording the changes as they are conducted during the design process or 2) comparing the two versions of the diagram. Table 1 compares both approaches, for the stated reasons we chose the second approach for our work.

As a motivation for our further progress, take a look at Figure 3. It contains two versions of a fragment of a PSM schema. In the old version, class `Item` has an attribute `price`, in the new version, class `Purchase` has an attribute `price`.

recording changes	**schema comparison**
the recorded set should be normalized to eliminate redundancies (repeated changes in the same place etc.).	No redundancies; the set of changes is always minimal.
Once the evolution process is started, the old version can not be easily changed.	Both versions and new can be edited without limitations.
Possibility to interrupt the work and continue in another session later. The sequence of recorded changes would have to be stored and recording resumed later.	The process of evolution can be arbitrarily stopped and resumed.
To retrieve the sequence for reverse process, the user will have to either start with the new version and record the operations needed to go back to the old version again, or the system will have to be able to create inverse sequence for each sequence of operations.	The reverse operation can be easily handled by the same algorithm, only with the two schemas on the input swapped.
For a schema from an outer source, the sequence of operation changes can not be retrieved directly; the user must start with the old version of the schema and manually adjust it to match the new schema.	A schema from an outer source can be imported[2] and serve as an input to the change detection algorithm.

Table 1. Approaches to Evolution: Recording Changes vs. Schema Comparison

Without any further information, there can be two interpretations of the change: 1) attribute `price` was moved from `Item` to `Purchase` or 2) the attribute was removed from `Item` and a new attribute was added to `Purchase`, coincidentally having the same name. Revalidation of the documents is even more difficult to decide. In the second case, a correct value must be assigned to the new attribute[3]. In the first case, the value of attribute `price` should be set to the sum of the values of `price` in all the items in the first version.

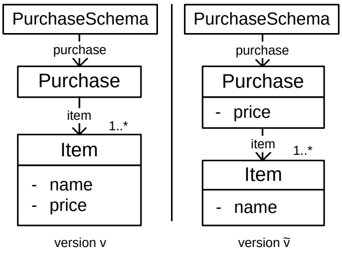


Fig. 3. Evolved PSM Schema

To achieve correct identification of “addition/removal” from “move” (as opposed to existing approaches), we need to extend the XSEM framework with a support for having multiple version in the model and possibility to link constructs in different versions.

Definition 3 (ver, version link, *getInVer*). *Let $\mathcal{V}$ be the set of versions for the model $\mathcal{M}$. Function $ver : \mathcal{M} \cup \mathcal{S}^{*\prime} \to \mathcal{V}$ returns the version a given construct belongs to. We require a natural condition of all the constructs of a PSM schema $\mathcal{S}'$ to belong to the same version, i.e. $(\forall \mathcal{S}' \in \mathcal{S}^{*\prime})(x \in \mathcal{S}'_{all} \leftrightarrow ver(x) = ver(\mathcal{S}'))$. An equivalence relation of* version links $\mathcal{VL} \subset \mathcal{M} \times \mathcal{M}$

[3] This is an open problem for our future work, see Section 7.

change	**description**
classAdded	new class was added to the schema
classRemoved	class was removed from the schema
classRenamed	existing class was renamed
classMoved	class is moved to the PSM tree
attributeAdded/Removed/Renamed	attribute was added/removed/renamed
attributeTypeChanged	attribute type changed
attributeIndexChanged	attributes were reordered in the class
attributeCardinalityChanged	cardinality of an attribute was changed
attributeMoved	attribute was moved from one class to another
associationAdded/Removed/Renamed	association was added/removed/renamed
associationIndexChanged	associations were reordered in the class
associationCardinalityChanged	cardinality of an attribute was changed
associationMoved	association is moved in the PSM tree

Table 2. Changes Identified in XSEM-Evo between two Versions of the same Schema

contains pairs of constructs that are different versions of the same construct. Partial function getInVer : $(\mathcal{M} \times \mathcal{V}) \rightarrow \mathcal{M}$, *getInVer*$(e, v) = \tilde{e} \leftrightarrow \exists\ \tilde{e} \in \mathcal{M}$: $(e, \tilde{e}) \in \mathcal{VL}\ \wedge\ ver(\tilde{e}) = v$ *returns a construct in a desired version (if* e *does not exist in version* v*, getInVer*(e, v) *is undefined). If* $(x, \tilde{x}) \in \mathcal{VL}$*,* x *and* $\tilde{x}$ *must both be constructs of the same kind (e.g. both classes, attributes, PSM associations...).*

In the following text, we will assume $|\mathcal{V}| = 2$, unless explicitly stated otherwise (i.e. we expect that there are two versions in the system - an old version ($v \in \mathcal{V}$) and a new version ($\tilde{v} \in \mathcal{V}$)). Values of *ver* function form the input of the change detection algorithm (and so does the relation $\mathcal{VL}$). In XCase, values of *ver* are assigned automatically by the system during and maintained As the user edits the schema (either the old or the new version).

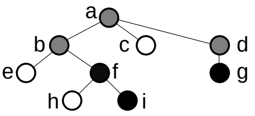
5 Changes

Based on Definitions 1 and 2, we can identify all possible changes that can occur between two versions. Each change is formally defined as a predicate with certain amount of parameters characterizing the change. Detecting changes means finding the constructs that match each respective predicate. Table 2 lists all recognized changes.

Describing each change and it's revalidation is beyond the scope of this paper. Therefore we will describe only *attributeMoved* change in detail (an example of this change is depictend in Figure 3), other changes are treated in a similar manner. Change *attributeMoved* means that the value of *class'*(A') is changed i.e. an attribute is moved from class C'_o to another PSM class $\widetilde{C'_n}$ (and the new index of $\tilde{A}'$ is $\tilde{i}'$). Formally *attributeMoved* is defined as a ternary predicate:

$$\begin{aligned}&\mathit{attributeMoved}(\tilde{A}', \widetilde{C'_n}, \tilde{i}') \leftrightarrow \tilde{A}' \in \widetilde{\mathcal{S}'_a} \wedge \widetilde{C'_n} \in \widetilde{\mathcal{S}'_c} \wedge \tilde{i}' \in \mathbb{N}_0 \wedge \mathit{getInVer}(\tilde{A}', v) \neq \mathit{null} \wedge \\ &\quad \mathit{class'}(\tilde{A}') = \widetilde{C'_n} \wedge (\exists C'_o \in \mathcal{S}'_c)(\mathit{class'}(\mathit{getInVer}(\tilde{A}', v)) = C'_o \wedge \mathit{getInVer}(\widetilde{C'_n}, v) \neq C'_o)\end{aligned}$$

Moving an attribute is an evolution operation that requires more in-depth enquiry. The aim of our approach is to keep the semantics of the revalidated document and not to lose the existing data during revalidation. The trivial solution – deleting the attribute from its former location in the document and creating a new attribute in the new location (as used in [6] and [5]) – is not suitable, because the value of the attribute is lost. The most general approach is to couple each instance $(\tilde{A}', \widetilde{C'_n}, \tilde{i}')$ of *attributeMoved* change with a revalidation function $attMove_{\tilde{A}'}(oldLocations,\ newLocation)$: $XSEMPath \times XSEMPath \rightarrow type'(\tilde{A}')$[4] where *oldLocations* is an expression returning all existing instances of A' and *newLocation* is an expression containing the path to one instance in the new document. The result of the function is the new value for the instance of $\tilde{A}'$ in the new document. In the general case the function $attMove_{\tilde{A}'}$ is defined by the user, but the system can give the user a suggestion in certain cases – several types of the most common scenarios can be distinguished.



The figure on the left shows an example of *tree* function, the result of $tree(f, g, i)$ is the set $\{a, b, d, f, g, i, a-b, b-f, f-i, a-d, d-g\}$, a is the root of the common subtree and $a-b$ the edge from a to b.

Fig. 4. Example for *tree* function

In the following text we expect that the attribute was moved between classes C'_o and $\widetilde{C'_n}$, $A' \in attributes'(C'_o)$, $\tilde{A}' \in attributes'(\widetilde{C'_n})$. Let $\widetilde{C'_o} = getInVer(C'_o, \tilde{v})$ and $C'_n = getInVer(\widetilde{C'_n}, v)$, be the new version of class C'_o and the old version of class $\widetilde{C'_n}$ respectively, both can be *null*. In the situation depicted in Figure 3 $A' = \texttt{price}_1, \tilde{A}' = \texttt{price}_2, C'_o = \texttt{Item}_1, C'_n = \texttt{Purchase}_1, \widetilde{C'_o} = \texttt{Item}_2, \widetilde{C'_n} = \texttt{Purchase}_2$ (we use subscripts to distinguish constructs in version v and $\tilde{v}$).

Let $tree(\mathcal{X}')$ return the subgraph of the smallest common subtree for a set $\mathcal{X}' \subseteq \mathcal{N}'$ containing the root B' of the common subtree, members of $\mathcal{X}'$ and for each $X' \in \mathcal{X}'$ path between X' and the root B' (see Figure 4 for an example). We define predicate $stable(\mathcal{X}') \leftrightarrow \mathcal{X}' \subseteq \{\mathcal{S}'_{all} \setminus \mathcal{S}'_a\} \wedge \{\forall X' \in \mathcal{X}' :$ X' is present in the new version, it was not moved, added or deleted and cardinality was not changed (if X' is an association)$\}$. The intuitive meaning of predicate $stable(\mathcal{X}')$ is that there were no radical changes in the structure for the members of $\mathcal{X}'$. If $C'_n \neq null, T' = tree(\{C'_o, C'_n\})$ and $stable(T')$ holds, than if:

[4] The formal definition of *XSEMPath* expressions is beyond the scope of this paper. Intuitively, the notation and usage is similar to *XPath* [15], only the *XSEMPath* expressions refer to the constructs from a PSM schema instead of the names of XML elements and attributes in the document. When executed upon a document $T \in \mathcal{T}(\mathcal{S}')$, *XSEMPath* query is translated to *XPath* query and the result of this query upon T is returned.

1. $\forall$ *association* $R' \in T'$: *card'*$(R') = m_{R'}..1$ (only cardinalities 0..1 and 1..1 are allowed in the affected part of the schema). In this simplest case the attribute $\tilde{A}'$ will have 0 or 1 instance in the old schema. Then this instance can be copied to the only one new location ($attMove_{\tilde{A}'}$ will be *identity* function).
2. C'_o is a descendant of C'_n in the PSM tree (the attribute is moved upwards, but the associations between C'_o and C'_n can have arbitrary cardinalities). Then all instances of A' under each instance of C'_n should be "aggregated" to one instance of $\tilde{A}'$ (this is the case depicted in Figure 3). Several aggregation functions can be offered (e.g. *sum, count, avg, max, min* known from relational databases or *concat* inlining the respective values).
3. C'_o is a descendant of C'_n in the PSM tree, *card'*$(A') = m'..1$ and *card'*$(\tilde{A}') = \widetilde{m'}..*$. Then this case is similar as the case above, but the cardinality of attribute $\tilde{A}'$ is adjusted, so all the values from existing instances can be used as values of $\tilde{A}'$, no aggregation is needed ($attMove_{\tilde{A}'}$ can be called *toList*).
4. C'_o is an ancestor of C'_n in the PSM tree, *card'*$(A') = m'..*$ and *card'*$(\tilde{A}') = \widetilde{m'}..1$. Then this is an inverse case to the one above. The respective values of A' can be distributed to the new locations. Nonetheless, a user may have to specify the distribution precisely ($attMove_{\tilde{A}'}$ can be called *distribute*).

When none of the conditions above is satisfied, a possible general approach is to use the function $attMove_{\tilde{A}'} = identityN$ which returns the value of the n-th instance of A' when required on the n-th location in the revalidated document. Other $attMove_{\tilde{A}'}$ functions must be entered by the user.

6 Revalidation

In the first step, the revalidation algorithm detects all changes between two versions of the model. If changes that require revalidation occur, a revalidation script is generated. Applying the script on a document valid against the old version produces a document valid against the new version. Since changes are defined as changes in the XSEM model, similarly as XSEM model can be translated to an XML schema in any XML schema language, the set of changes can be used to generate a revalidation script in any kind of implementation language.

Example 3. The in-depth description of the process of generating the revalidation script is beyond the scope of this paper. As an example, we will show the revalidation script in XSL for the schema depicted in Figure 3, concretely if the first interpretation from Example 4. In that case, we have to revalidate the *attributeMoved* change. In this example, we will use the subscript 1 for constructs from the old version and 2 for constructs from the new version. We are revalidating *attributeMoved*$(\texttt{price}_2, \texttt{Purchase}_2, 0)$. Since there are no other changes, *stable*$(\{\texttt{Purchase}_1, \texttt{Item}_1\})$ also holds and since the association `Purchase-Item` has cardinality $1..*$ and `Purchase` is a parent of `Item`, this case belongs to the second group mentioned on page 9 during the description of the revalidation of *attributeMoved* change (aggregation upwards). Before generating the script,

The XSL revalidation script consists of four templates. The first template processes `purchase` element – it creates the wrapping tag and calls a template for the moved PSM attribute `pur-price` and for `item` elements.
The second template processes item elements (the template is necessary, because we need to remove the `price` subelement from `item` elements).
The fourth template copies the content of each `name` element to the output.
The third, named, template `pur-price` calls the *attMove*$_{\texttt{price}_1}$ function (the *XSEMPath* expressions are converted to regular *XPath* expressions – *oldLocations* to `‘item/price’` and *newLocation* to `‘price’`). The user selected the function *sum* as *attMove*$_{\texttt{price}_1}$, a generic function built in the system. If none of the built-in function is suitable for the situation, the user can extend the system with a custom set of functions (having the same header as *sum*).

```
<xsl:stylesheet>
  <xsl:template match=‘/purchase’>
    <purchase>
      <xsl:call-template
        name=”pur-price ’ />
      <xsl:apply-templates
        select=‘item ’ />
    </purchase>
  </xsl:template>
  <xsl:template
    match=‘/purchase/item’>
    <item>
      <xsl:copy-of select=‘name’ />
    </item>
  </xsl:template>
  <xsl:template name=‘pur-price’>
    <price>
      <xsl:value-of select=
        ‘xc:sum(item/price,
                price)’/>
    </price>
  </xsl:template>
  <xsl:template
    match=‘/purchase/item/name’>
    <xsl:copy-of select=‘.’ />
  </xsl:template>
  <xsl:function name=‘xc:sum’ >
    <xsl:param
      name=‘oldLocations ’ />
    <xsl:param
      name=‘newLocation ’ />
    <xsl:value-of select=
      ‘sum($oldLocations)’ />
  </xsl:function>
</xsl:stylesheet>
```

Fig. 5. Example of a Revalidation Script in XSL

the user selects the function *sum* to be used as *attMove*$_{\texttt{price}_1}$. The resulting revalidation script in XSL is depicted and explained in Figure 5.

7 Conclusion and Future Work

In this paper we proposed a formal model for schema evolution. We formally defined changes that can be detected between two versions of a schema. We sketched an implementation of an algorithm that produces revalidation script on the basis of the detected set of changes.

The set of the detected changes is useful not only for the revalidation algorithm, but it is also possible to immediately decide, whether revalidation is needed or not and help the user to locate the changes in the schema.

The revalidation script can deal with structural modifications and with a significant part of the content modifications automatically, user input is required only where necessary (e.g. when a new content must be added during revalidation). We plan to extend its capabilities in adding content in our future work.

The algorithm in its current version deals mainly with revalidation of 1) structure and 2) data already present in the document. Because new data are often required for new versions, we will focus our future work on obtaining this data for the revalidated documents. For this purpose, we will utilize the existing connection between PIM and PSM and and a new similar connection between PIM and the model of a data storage (e.g. an ER schema [13]).

The change detection algorithm requires semantic links between the two compared versions of the schema, which are not available for schemas were imported and not created with an XSEM-enabled tool. Existing approaches to schema comparison and matching can be utilized as heuristics for creating these links.

Type inheritance is a fundamental part of large specifications, but support for inheritance is limited in the current version of XSEM. We intend to extend both PIM and PSM levels with inheritance support.

References

1. XCase – tool for XML data modeling. `http://xcase.codeplex.com/`.
2. *MOF 2.0 / XMI Mapping Specification, v2.1.1.* OMG, 2009.
3. R. Bourret. XML and Databases. September 2005. `http://www.rpbourret.com/xml/XMLAndDatabases.htm`.
4. D. Booth, C. K. Liu. *Web Services Description Language (WSDL) Version 2.0 Part 0: Primer.* W3C, June 2007. `http://www.w3.org/TR/wsdl20-primer/`.
5. G. Guerrini, M. Mesiti, and M. A. Sorrenti. Xml schema evolution: Incremental validation and efficient document adaptation. In *XSym*, volume 4704 of *Lecture Notes in Computer Science*, pages 92–106. Springer, 2007.
6. Hong Su, D. K. Kramer, E. A. Rundensteiner. XEM: XML Evolution Management, Technical Report WPI-CS-TR-02-09. 2002.
7. M. Klettke. Conceptual xml schema evolution — the codex approach for design and redesign. In *Workshop Proceedings Datenbanksysteme in Business, Technologie und Web (BTW 2007)*, pages 53–63, Aachen, Germany, March 2007.
8. J. Miller and J. Mukerji. *MDA Guide Version 1.0.1.* Object Management Group, 2003. http://www.omg.org/docs/omg/03-06-01.pdf.
9. M. Necasky. *Conceptual Modeling for XML*, volume 99 of *Dissertations in Database and Information Systems Series.* IOS Press/AKA Verlag, January 2009.
10. M. Necaský and I. Mlýnková. Five-level multi-application schema evolution. In *DATESO*, pages 90–104, 2009.
11. Object Management Group. *UML 2.1.2 Specification*, nov 2007. `http://www.omg.org/spec/UML/2.1.2/`.
12. T. Bray, J. Paoli, C. M. Sperberg-McQueen, E. Maler, F. Yergeau. *Extensible Markup Language (XML) 1.0 (Fifth Edition).* W3C, November 2008. `http://www.w3.org/TR/2008/REC-xml-20081126/`.
13. B. Thalheim. *Entity-Relationship Modeling: Foundations of Database Technology.* Springer Verlag, Berlin, Germany, 2000.
14. H. S. Thompson, D. Beech, M. Maloney, and N. Mendelsohn. *XML Schema Part 1: Structures (Second Edition).* W3C, October 2004. `http://www.w3.org/TR/xmlschema-1/`.
15. W3C. *XML Path Language (XPath) 2.0.* `http://www.w3.org/TR/xpath20/`.

XML Document Correction and XQuery Analysis with *Analyzer*

Jakub Stárka, Martin Svoboda, Jiří Schejbal, Irena Mlýnková, and David Bednárek

Department of Software Engineering, Faculty of Mathematics and Physics
Charles University in Prague
Malostranské náměstí 25, 118 00 Praha 1, Czech Republic
{starka, svoboda, mlynkova, bednarek}@ksi.mff.cuni.cz

Abstract. This paper describes extensions of our previously proposed SW prototype – Analyzer, a framework for performing statistical analyses of real-world XML data. Firstly, it describes the design and implementation of a system for the analysis of collection of XQuery programs. It is based on the frequency of the occurrence of various language constructs and their combinations defined by the user. In the core of the system, the XQuery program is converted to a suitable XML representation which allows for analytical queries formulated in the XPath language. Secondly, we introduce the model involving repairs of elements and attributes with respect to single-type tree grammars. Via the inspection of the state space of an automaton recognising regular expressions, we are always able to find all minimal repairs represented by recursively nested multigraphs, which can be translated to particular sequences of edit operations altering data trees. We have proposed four particular algorithms and provided the prototype implementation supplemented with experimental results.

1 Introduction

The eXtensible Markup Language (XML) [5] is currently a de-facto standard for data representation. Its popularity is given by the fact that it is well-defined, easy-to-use and, at the same time, enough powerful. The problem is that the XML standards were proposed in full possible generality so that future users can choose what suits them most. Nevertheless, the real-world XML data are usually not so “rich”, thus the effort spent on every possible feature is mostly useless.

Exploitation of results of statistical analyses of real-world data is a classical optimization strategy in various areas of data processing. It is based on the idea to focus primarily on efficient implementation of constructs that are used in real-world data most often. One of the most important advantages of statistical analyses of real-world data is refutation of incorrect assumptions on typical use cases, features of the data, their complexity etc. As an example we can consider exploitation of recursion. The support for recursion is often neglected and it is considered as a side/auxiliary construct. However, analyses [11] show that in

V. Snášel, J. Pokorný, K. Richta (Eds.): Dateso 2011, pp. 61–72, ISBN 978-80-248-2391-1.

selected types of XML data it is used quite often and, hence, is efficient, or at least any support is very important. On the other hand, the number of distinct recursive elements is typically low (for each category less than 5) and that the type of recursion commonly used is very simple.

However, working with real-world data is not simple, because they can often change, are not precise, or even involve a number of errors. In this case we can either discard the incorrect data, and, hence, loose a significant portion of them, or provide a kind of *corrector*.

In the next step we want to make the analyses themselves. Currently there exists a number of papers which focus on statistical analyses of real-world XML data [11, 3, 10], however an analysis of real-world XML operations, in particular queries, is still missing. The reason is mainly the complexity of crawling a representative set and the complexity of the analytical process.

In this paper we describe two parts of a general framework for statistical analyses of real-world XML data called *Analyzer*. Firstly, we focus on a correction framework involving structural repairs of elements with respect to a single-type tree grammar. Secondly, we describe the usage of the framework for XQuery analysis. Since there is no standardized real-world datasets, we use two artifical collections to demonstrate the approach.

Outline The paper is structured as follows: In Section 2 we describe the related work, in particular concerning corrections of data. In Section 3, we describe the architecture of *Analyzer* which indicates its general functionality. Section 4 is devoted to processing of incorrect data. In Section 5, we show the principles of used query analysis and we show results of a query analysis of some artificial data. Finally, in Section 6 we conclude.

Relation to Previous Work In this paper, we extend our previous work [17]. Motivated by a successful and interesting statistical analysis of real-world XML data [11], *Analyzer* was implemented as a SW project of Master students of the Department of Software Engineering of the Charles University in Prague. Its installation package as well as documentation and source files can be found at its official web site [14]. Its first release 1.0 involved only basic functionality to demonstrate its key features and advantages and it was briefly introduced in paper [17]. In the following text, we describe extensions of *Analyzer* focused on XML document correction which was firstly proposed in [15] and extended in [16], and new module for XQuery analysis.

2 Related Work

As we have mentioned in the introduction, while currently there exists a number of papers focussing on statistical analysis of XML documents, XML schemas or their mutual comparison [11, 3, 10], there is no paper that would describe either the results or the process of analysis of real-world XML queries. Thus in this section we focus on the the related work of the second aim of this paper – correction of XML documents, in particular their re-validation.

The proposed correction model is based primarily on ideas from [2] and [13]. Authors of the former paper dynamically inspect the state space of a finite automaton for recognising regular expressions in order to find valid sequences of child nodes with minimal distance. However, this traversal is not effective, requires a threshold pruning to cope with potentially infinite trees, repeatedly computes the same repairs and acts efficiently only in the context of incremental validation. Although these disadvantages are partially handled in the latter paper, its authors focused on documents querying, but not repairing.

Next, we can mention an approximate validation and correction approach [18] based on testers and correctors from the theory of program verification. Repairs of data inconsistencies like functional dependencies, keys and multivalued dependencies are the subject of [12, 20].

Contrary to all existing approaches, we consider single type tree grammars instead only local tree grammars. Thus, we work both with DTD and XML Schema. Approaches in [2, 18] are not able to find repairs of more damaged documents, we are able to always find all minimal repairs and even without any threshold pruning to handle potentially infinite XML trees. Next, we have proposed much more efficient algorithm following only perspective ways of the correction and without any repeated repair computations. Finally, we have a prototype implementation available at [7] and performed experiments show a linear time complexity depending on a number of nodes in documents.

3 Framework Description

This section briefly concerns with *Analyzer* architecture, proposed analyses model and basic implementation aspects. The details are described in paper [17].

Architecture. The *Analyzer* allows to work with multiple opened projects at once, each representing one analytical research intent. Thus, we can divide the framework architecture into two separate levels, as it is depicted in Figure 1. The first one contains components, which are shared by all these projects. The second one represents components exclusively used and created in each opened project separately (repositories, storages, crawlers and entity managers).

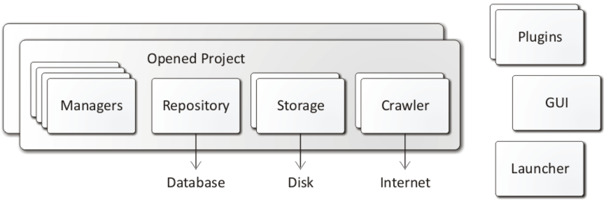


Fig. 1. Analyzer Framework Architecture

Repositories serve for storing all computed analytical data and the majority of project configuration metadata. Storages are used for storing document contents, i.e. binary contents of analyzed files. Finally, documents to be analyzed can be inserted into existing projects through import sessions (locally accessible files) or download sessions (downloading files from the Internet via crawlers).

The project layer contains a set of managers, which are responsible for creating, editing and processing of all analysis entities such as documents, collections of documents or reports over collections. As all computed analytical data are stored permanently in a repository, in order to increase efficiency, these managers are able to cache loaded data and some of them even to postpone and aggregate required update operations without violating the consistency.

Analyses. Although the framework enables also more complex usages, the standard life cycle of each project can be represented by the following phases.

1. Creation of a new project and configuration of its components,
2. Selection and configuration of analyses using available plugins,
3. Insertion of documents to be analyzed through import or download sessions,
4. Computation of analytical results over documents of a given relative age,
5. Selection and configuration of collections and clusters of them,
6. Document classification and assignment into collections, and
7. Computation of final statistical reports over particular collections.

Plugins. *Analyzer* itself provides a general environment for performing analyses over documents and collections of documents, but the actual analytical logic is not a part of it. All analytical computations and mechanisms are implemented in plugins. Not only that each particular plugin is intended only for processing of selected document types only, the user is also able to configure available plugins and, thus, adjust their behaviour to desired analytical intents.

The plugin functionality itself is provided through implemented methods, which are of eight predefined types listed in the following enumeration.

- The *detector* recognizes types of a processed document,
- The *tracer* looks for outgoing links in a given document,
- The *corrector* attempts to repair a content of a given document,
- The *analyzer* produces results over a given document,
- The *collector* classifies documents into collections of a given cluster,
- The *provider* creates reports over documents in a collection,
- The *viewer* serves for browsing computed results over a document, and
- The *performer* serves for browsing computed reports over a collection.

4 Processing of Incorrect Data

During the phase of document processing and result generating in *Analyzer* framework, *corrector* methods of available plugins are able to modify data contents of such documents. This feature is motivated primarily by the possibility of working with potentially incorrect documents.

In this section, we particularly focus on the problem of structural invalidity of XML documents. In other words, we assume the inspected documents are well-formed and constitute trees, however, these trees do not conform to a schema in

DTD [5] or XML Schema [9]. Having a potentially invalid XML document, we process it from its root node towards leaves and propose minimal corrections of elements in order to achieve a valid document close to the original one.

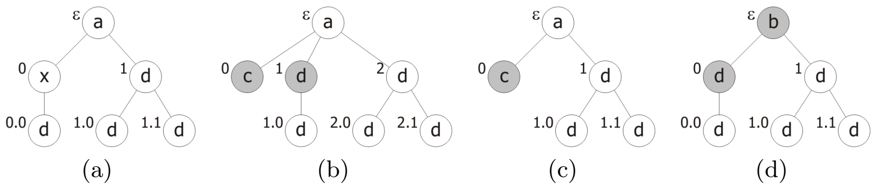


Fig. 2. Sample invalid XML tree with three possible minimal repairs

In Figure 2 we can see a sample correction process. Item (a) represents an original XML document, where element names are depicted by labels inside nodes. Without any detailed schema knowledge, assume only that element x at position 0 is not allowed. Processing this invalid tree, our algorithm finds three different minimal repairs, all of which are outlined in Items (b), (c) and (d).

The remaining parts of this section will present basic ideas of our correction model and proposed algorithms for finding structural repairs of invalid XML documents. Details of this proposal are presented in [16, 15].

4.1 Proposed Solution

Our correction framework is capable to generate local structural repairs for invalid elements. These repairs are motivated by the classic Levenshtein metric for strings. For each node in a given XML tree and its sequence of child nodes we attempt to efficiently inspect new sequences that are allowed by the corresponding content model and that can be derived using the extended concept of measuring distances between strings. However, in our case we do not handle only ordinary strings, but sequences derived from node labels with nested subtrees.

The correction algorithm starts processing at the root node and recursively moves towards leaf nodes. We assume that we have the complete data tree loaded into the system memory and, therefore, we have a direct access to all its parts. Under all conditions the algorithm is able to find all minimal repairs, i.e. repairs with the minimal distance to the grammar and the original data tree according to the introduced cost function.

Edit Operations. Edit operations are elementary transformations that are used for altering invalid data trees into valid ones. They behave as functions, performing small local modifications with a provided data tree. Despite the correction algorithm does not directly generate sequences of these edit operations, we can, in the end, acquire them using a translation of generated repairs, as it will be explained later. We have proposed and implemented edit operations capable to insert a new leaf node, delete an existing one and rename a label of a node.

Edit operations can be composed together into sequences. And if these sequences fulfil certain qualities, they can be classified as update operations. In this way we can work with update operations capable to insert a new subtree, delete an existing subtree and recursively repair a subtree with an option of changing a label of its root node.

Repairing Instructions. Assume that we are in a particular node in a data tree and our goal is to locally correct this node by correcting the sequence of its child nodes. Since the introduced model for measuring distances uses only non-negative values for the cost function, in order to acquire the global optimum, we can simply find minimal combinations of local optimums, i.e. minimal repairs for all subtrees of original child nodes of the inspected one.

However, we need to find all minimal repairs and even represent them in a compact repair structure. For this purpose we use repairing instructions. We have exactly one instruction for each edit operation and these instructions represent the same transformation ideas, however, do not include particular positions to be applied on. Having a sequence of instructions at a given level, we can easily translate it into all corresponding sequences of edit operations later on.

Correction Intents. Being in a particular node and repairing its sequence of child nodes, the correction algorithm generally has many ways to achieve the local validity proposing repairs for all involved child nodes. As already outlined, these actions follow the model of measuring distances between ordinary strings. The Levenshtein metric is defined as the minimal number of required elementary operations to transform one string into another.

We follow the same model, however, we have edit and update operations respectively and sequences of nodes. For example, an insertion of a new subtree at a given position stands for the insertion of its label into the corresponding node sequence and, of course, recursive generation of such new subtree.

The algorithm attempts to examine all suitable new words that are in the language of the provided regular expression restraining the content model of the inspected parent node. We do not generate word by word, but we inspect all suitable words statically using a notion of a correction multigraph. Correction intents represent assignments for these multigraphs, i.e. the recursive data tree processing in a top-down manner.

Correction Multigraphs. All existing correction intents in a context of a given parent node can be modelled using a multigraph for this node. Vertices of a multigraph for n child nodes can be divided into $n+1$ disjoint strata, vertices of each stratum correspond to states of the Glushkov automaton for recognising the provided regular expression. Edges in a multigraph are derived from the automaton transition function and they represent nested correction intents.

In order to find best repairs for a provided sequence of nodes, we need to find all shortest paths in the multigraph. Therefore, we first need all its edges to be associated with already evaluated nested repair structures and their minimal costs. And this represents nontrivial nested recursive computations. Anyway, we require that each edge can be evaluated in a finite time, otherwise we would obviously not be able to find required shortest paths at all.

Repairs Construction. Each correction intent can essentially be viewed as an assignment to the nested recursive processing. The correction of a provided data tree is initiated as a special starting correction intent for the root node and processing of every intent always involves the construction of at least the required part of the introduced multigraph with other nested intents.

Therefore, we continuously invoke recursive computations of nested intents. When we reach the bottom of the recursion, we start backtracking, i.e. encapsulating all found shortest paths into a form of a compact repair structure and, then, passing it one level up, towards the starting correction intent.

Having constructed a repair structure for the starting intent, we have found corrections for the entire data tree. Each intent repair contains encoded shortest paths and related repairing instructions. Now we need to generate all particular sequences of repairing instructions and translate them into standard sequences of edit operations. Having one such edit sequence, we can apply it on the original data tree and we obtain its valid correction with a minimal distance.

Correction Algorithms. Now, we have completely outlined the proposed correction model. However, there are several related efficiency problems that would cause significantly slow behaviour, if we would strictly follow this model. Therefore, we have introduced two particular correction algorithms, which are described in detail in [16]. They both produce the same repairs, but there are differences in their efficiency.

The first algorithm is able to directly search for shortest paths inside each intent computation and, therefore, does not need the entire multigraphs to be constructed. The next improvement is based on caching already computed repairs using signatures distinguishing different correction intents, but intents with the same resulting repair structure. This causes that this algorithm never computes the same repair twice. The second algorithm is able to compute lazily even to the depth of the recursion. We have achieved this behaviour by scattering all nested intents invocation and multigraph edges evaluation into small tasks, which are incrementally executed by a simple scheduler.

5 Query Analysis

In this section, we describe the second unique feature of *Analyzer* – *XQAnalyzer* – a tool designed to support studies that include analysis of a collection of XQuery programs. *XQAnalyzer* consumes a set of XQuery programs, converts them into a kind of intermediate code, and stores this internal representation in a repository. Subsequently, various analytical queries may be placed on the repository to determine the presence or frequency of various language constructs in the collection, including complex queries focused on particular combinations of constructs or classes of constructs.

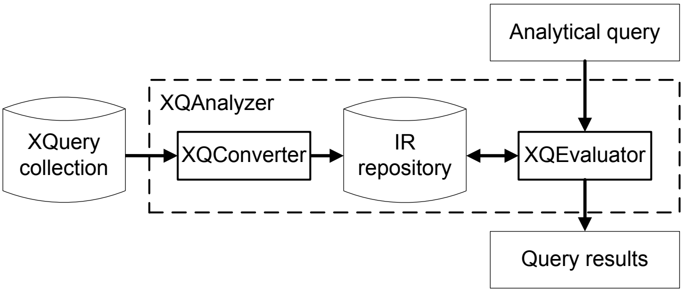


Fig. 3. The architecture of the *XQAnalyzer*

The architecture of the *XQAnalyzer* is shown in Figure 3. Each document from a given collection of XQuery programs is parsed and converted to the internal representation by the *XQConverter* component. The *XQEvaluator* component evaluates analytical queries and returns statistical results.

5.1 Analytical Queries

In the *XQAnalyzer*, the term *analytical query* denotes a pattern or condition placed on a XQuery program, usually a search for a feature. Each XQuery program in the collection is evaluated independently, producing either a boolean value or a hit count representing the presence or number of occurrences in the program, respectively. The *XQEvaluator* then returns various statistical results like the percentage of programs which contain the searched feature or the histogram of hit counts over the repository.

Given the fact that the tool is designed for research in the area of XML and, in particular, XQuery, the best choice would be a query language derived from XPath. XPath is naturally well-known in the community and it is designed to place pattern-like queries on tree structures – in our case, a tree is a typical representation of a program during early stages of its analysis.

5.2 Internal Representation of XQuery Programs

The key issue in the design of *XQAnalyzer* is the internal representation of XQuery programs. In our approach, we do not want to limit the nature of the analytical queries; therefore, the internal representation must store any XQuery program without loss of any feature (perhaps except of comments). Furthermore, the internal representation is exposed to the user via the query interface; therefore, it should be as simple as possible. Finally, the internal representation affects the performance of the *XQEvaluator*.

The W3C standards related to XQuery define at least the following two formalisms that might be used as a base for our internal representation:

- The *XQuery Grammar* (in Extended Backus-Naur Form) defined in [4].
- The *Normalized XQuery Core Grammar* (also in EBNF) defined in [8].

Note that the XQuery formal semantics [8] is defined in terms of static/dynamic evaluation rules that may be considered a kind of internal representation too. However, their application in our analytical environment would be impractically difficult.

Among the existing formalisms mentioned so far, we have chosen the Normalized XQuery Core Grammar. There are the following reasons behind this decision:

- It is a part of the standard, therefore well known and not skewed towards any evaluation strategy.
- It is smaller than the full XQuery Grammar and it hides the redundant features of the XQuery language.

The final set of nonterminals is listed in Tab. 1 together with their frequency in selected collections of XQuery programs (see Sec. 5.3). When our internal representation is presented in the form of a XML document, these nonterminals become *XML elements.*

The rest of the semantic information is enclosed in *XML attributes* attached to these elements. These attributes contain either data extracted from the source text (like names of variables or contents of literals) or additional semantic information (like the axis used in an XPath axis step). In addition to these data required to preserve the semantics, we also added attributes that may help recovering the original syntax before the normalization to XQuery Core (e.g. whether the abbreviated or the full syntax was used in axis step).

5.3 Results

Since there is no standardized collection of real-life XQuery programs yet (except of small benchmarks like XMark [1]), we have chosen two artificial collections associated to the W3C XQuery language specification: The XQuery Use Cases [6] and the XQuery Test Suite [19]. The Use Cases collection consists of 85 “text-book” XQuery programs prepared to demonstrate the most important features of the language, the Test Suite collection contains 14869 small XQuery programs created to cover all features (the remaining 252 files in the original collection contain intentional parse errors). Although the Test Suite collection is more than 100 times larger in terms of the number of files, the real ratio of sizes (in terms of the number of AST nodes) is 31:1 because the Use Cases files are larger.

In Tab. 1 we show the frequency of core elements of the language, named accordingly to the abstract grammar nonterminals derived from the Normalized

XQuery Core Grammar (see Sec. 5.2). The percentages are defined by the number of occurences divided by the total number of abstract syntax tree nodes in the collection (which was 4 469 for the Use Cases and 138 949 for the Test Suite).

Besides the obvious difference between the two collections, corresponding to their purpose, there are the following noticeable observations: The frequency of quantified expressions (`some` or `every`) is about eight times smaller than the frequency of `for`-expression. The `if`-expression is quite rare – once per 30 `for`-expressions or 50 operators. A number of features like `ordered`/`unordered`-expressions are omitted in the Use Cases. While frequent in the Test Suite, the comma operator is surprisingly rare in the Use Cases.

Table 2 shows the use of the twelve XPath axes. The percentages represent the frequency of individual axes among all axis step operators in the collection (which was 638 for the Use Cases and 6 623 for the Test Suite). Notice that the results correspond to the traditional belief that many axes are extremely rare.

6 Conclusion

The main aim of this paper was to describe several research problems related to a complex extensible framework for analyses of real-world XML data called *Analyzer*. Firstly, we have proposed a correction framework dealing with invalid nesting of elements in XML documents using the top-down recursive processing of potentially invalid data trees. Contrary to existing approaches, we have considered the class of single type tree grammars instead only local tree grammars. We are able to find all minimal repairs. Secondly, we described *XQAnalyzer* and implemented a tool for analysis of collections of XQuery programs. *XQAnalyzer* works with a set of XQuery programs and translates them into an intermediate code. Subsequently, analytical queries may be placed over these translations to get the precence of quantity of specific constructs.

In our future plans, we will focus on further improvements of existing plugins related to XML data analyses and their exploitation in throughout analysis of both current state of real-world XML documents and evolution of XML data in the following months. We plan to repeat the analysis monthly and publish the new as well as aggregated results on the Web. We believe that such a unique analysis will provide the research community with important results useful for both optimization purposes as well as development of brand new approaches. Concurrently, we will shift our target area to the new types of data such as RDF triples, linked data, ontologies etc.

Acknowledgement

This work was partially supported by the Czech Science Foundation (GAČR), grants number 201/09/P364 and P202/10/0573.

Element	Use Cases	Test Suite
AtomicType	0.27%	2.49%
Axis	14.28%	4.79%
BaseURIDecl	—	0.04%
BindingSequence	3.62%	1.11%
BoundarySpaceDecl	—	0.07%
CData	—	0.01%
CaseClauses	0.02%	0.03%
CharRef	—	0.02%
CommaOperator	0.04%	2.04%
ConstructionDecl	—	0.04%
Constructor	3.36%	2.07%
Content	4.03%	2.11%
ContextItem	0.22%	0.11%
CopyNamespacesDecl	—	0.02%
DefaultCase	0.02%	0.03%
DefaultCollationDecl	—	0.01%
DefaultNamespaceDecl	—	0.12%
ElseExpression	0.07%	0.08%
EmptyOrderDecl	—	0.03%
EmptySequence	0.02%	0.63%
EntityRef	—	0.01%
Extension	—	0.03%
FLWOR	1.97%	0.79%
ForClause	2.10%	0.59%
FunctionBody	0.40%	0.16%
FunctionCall	5.77%	17.11%
FunctionDecl	0.40%	0.16%
Hint	0.38%	0.01%
IfExpr	0.07%	0.08%
InClauses	0.27%	0.15%
KindTest	4.83%	0.80%
LetClause	1.25%	0.32%
Literal	4.61%	20.32%
ModuleDecl	0.07%	0.00%
ModuleImport	0.07%	0.03%
Name	4.21%	2.14%
NameTest	10.14%	4.08%
NamespaceDecl	0.20%	0.18%
OperandExpression	0.02%	0.03%
Operator	3.85%	8.43%
OptionDecl	—	0.01%
OrderedExpr	—	0.01%
OrderingModeDecl	—	0.02%
Path	10.02%	2.51%
PragmaList	—	0.03%
QuantifiedExpr	0.27%	0.15%
QueryBody	1.83%	10.70%
ReturnClause	2.04%	0.90%
SchemaImport	0.38%	0.17%
String	6.82%	2.12%
TestExpression	0.34%	0.23%
ThenExpression	0.07%	0.08%
TupleStream	1.97%	0.79%
Type	0.98%	2.62%
Typeswitch	0.02%	0.03%
UnorderedExpr	—	0.01%
ValidateExpr	—	0.02%
VarDecl	—	2.42%
VarRef	8.68%	3.47%
VarValue	—	2.42%

Table 1. The elements of the internal representation

Element	Use Cases	Test Suite
child	71.63%	82.67%
descendant	—	0.21%
attribute	5.33%	3.70%
self	—	0.36%
descendant-or-self	23.04%	10.40%
following-sibling	—	—
following	—	0.44%
parent	—	0.50%
ancestor	—	0.44%
preceding-sibling	—	0.42%
preceding	—	0.42%
ancestor-or-self	—	0.44%

Table 2. Axis usage

References

1. Afanasiev, L., Marx, M.: An analysis of xquery benchmarks. Inf. Syst. 33(2), 155–181 (2008)
2. B. Bouchou, A. Cheriat, M. H. Ferrari Alves, A. Savary: Integrating Correction into Incremental Validation. In: BDA (2006)
3. Bex, G.J., Neven, F., den Bussche, J.V.: Dtds versus xml schema: a practical study. In: WebDB '04. pp. 79–84. ACM, New York, NY, USA (2004)
4. Boag, S., Chamberlin, D., Fernndez, M.F., Florescu, D., Robie, J., Simon, J.: XQuery 1.0: An XML Query Language (Second Edition). W3C (December 2010), `http://www.w3.org/TR/xquery/`
5. Bray, T., Paoli, J., Sperberg-McQueen, C.M., Maler, E., Yergeau, F.: Extensible Markup Language (XML) 1.0 (Fifth Edition). W3C (November 2008), `http://www.w3.org/TR/xml/`
6. Chamberlin, D., Fankhauser, P., Florescu, D., Marchiori, M., Robie, J.: XML Query Use Cases. W3C (March 2007)
7. Corrector Prototype Implementation, `http://www.ksi.mff.cuni.cz/~svoboda/`
8. Draper, D., Fankhauser, P., Fernández, M., Malhotra, A., Rose, K., Rys, M., Siméon, J., Wadler, P.: XQuery 1.0 and XPath 2.0 Formal Semantics. W3C (January 2007)
9. Gao, S., Sperberg-McQueen, C.M., Thompson, H.S.: W3C XML Schema Definition Language (XSD) 1.1 Part 1: Structures. W3C (December 2009), `http://www.w3.org/TR/xmlschema11-1/`
10. Mignet, L., Barbosa, D., Veltri, P.: The xml web: a first study. In: Proceedings of the 12th international conference on World Wide Web. pp. 500–510. WWW '03, ACM, New York, NY, USA (2003), `http://doi.acm.org/10.1145/775152.775223`
11. Mlynkova, I., Toman, K., Pokorny, J.: Statistical analysis of real xml data collections. In: COMAD'06. pp. 20–31. Tata McGraw-Hill Publishing, New Delhi, India (2006)
12. S. Flesca, F. Furfaro, S. Greco, E. Zumpano: Querying and Repairing Inconsistent XML Data. In: WISE '05. LNCS, vol. 3806/2005, pp. 175–188. Springer (2005)
13. S. Staworko, J. Chomicky: Validity-Sensitive Querying of XML Databases. In: Current Trends in Database Technology EDBT 2006, DataX06. Lecture Notes in Computer Science, vol. 4254/2006, pp. 164–177. Springer (2006)
14. Stárka, J., Svoboda, M., Sochna, J., Schejbal, J.: Analyzer 1.0. `http://analyzer.kenai.com/`
15. Svoboda, M.: Processing of Incorrect XML Data. Master Thesis, Charles University in Prague, Czech Republic (September 2010), `http://www.ksi.mff.cuni.cz/~mlynkova/dp/Svoboda.pdf`
16. Svoboda, M., Mlýnkova, I.: Correction of Invalid XML Documents with Respect to Single Type Tree Grammars. In: NDT 2011. Communications in Computer and Information Science, vol. 136. Springer, Macau, China (2011), [to be published]
17. Svoboda, M., Stárka, J., Sochna, J., Schejbal, J., Mlýnkova, I.: Analyzer: A framework for file analysis. In: BenchmarX '10. pp. 227–238. Springer-Verlag, Tsukuba, Japan (2010), `http://www.springerlink.com/content/078819t6645j6268/`
18. U. Boobna, M. de Rougemont: Correctors for XML Data. In: Database and XML Technologies. LNCS, vol. 3186/2004, pp. 69–96. Springer (2004)
19. W3C: XML Query Test Suite (November 2006)
20. Z. Tan, Z. Zhang, W. Wang, B. Shi: Computing Repairs for Inconsistent XML Document Using Chase. In: Advances in Data and Web Management. LNCS, vol. 4505/2007, pp. 293–304. Springer (2007)

Supporting Language Interoperability by Dynamically Switched Behaviors*

Jan Kurš[1], Jan Vraný[1], and Alexandre Bergel[2]

[1]Software Engineering Group,
Faculty of Informatics,
Czech Technical Universtity in Prague
{kursjan, jan.vrany}@fit.cvut.cz

[2]Pleiad Lab, Department of Computer Science (DCC)
University of Chile, Chile
http://bergel.eu

Abstract. Software programs are often written in more than one programming language as the emergence of domain specific languages testifies. Language interpreters are easily embeddable and performances are usually satisfactory. However, inter-language interaction remains a field tarnished by poor performances. The reason is that alien objects are wrapped, implying the use of expensive forwarding and converting mechanism.
We propose to represent alien objects as the set of different states and behaviors it may have by moving between languages, thus avoiding wrapping and conversion. We have validated our solution on integration of Java and Smalltalk programming languages.

Keywords: Programming Language, Virtual Machine, Object Transitions, Java, Smalltalk

1 Introduction

The last decade has seen the advent of domain specific languages and support for multi languages. Common execution platforms, including the JVM and .Net, are nowadays fit to execute programs written in more than one programming language. Whereas the execution mechanism needed to interpret these languages are fairly well accepted [7, 10], the way languages interact and exchanges values still remains an open topic.

The large majority of embedded languages convert or wrap objects when they cross the language boundary [12]. When an object is passed from one language interpreter to another, it is either converted or wrapped: values like integers, floats, booleans, characters, and strings are merely converted while the remaining objects are simply wrapped.

Whereas objects conversion and wrapping is a globally accepted among domain specific languages and scripting languages, it is the source of several problems and limitations. Consider a plain Java dictionary produced by a Java program. This dictionary is represented as an instance of java.util.HashMap. A JRuby interpreter will consider this

* This paper was partially supported by internal CTU grant – SGS 2011

object as a wrapped alien object. All calls done on this object implies a conversion or wrapping of its arguments and a delegation by the wrapper to the real objects. Sending the JRuby message put("One", 1) to the Java dictionary converts the JRuby string "One" and the JRuby integer 1 into their corresponding Java values. Delegating messages has a cost which is significant when intensively use.

A second problem is about object identity. When this Java dictionary is passed a second time to JRuby, it has to be wrapped using the same wrapper that was used for the first time. A bijective mapping between alien objects and wrappers has to be enforced. The wrapper used the second time has to be physically the same than the first wrapper (i.e., having the same pointer). Again, this comes at a fairly high cost in case of intensive object passing.

Instead of representing aliens objects as a wrapper in the host language, we propose to extend the definition of an object as a set of contextualized variable layouts and behaviour definition.

The proposed approach has been validated on Smalltalk/X programming environment that runs code in Smalltalk and Java programming languages. We have modified metaobject protocols in Smalltalk/X in order to implement proposed approach efficiently.

The contributions of this paper are: identification of the problems associated with objects crossing the language boundary; introduction of a new approach of moving objects between languages; description of a prototype implementation. The paper is organized as follows: The Section 2 introduces simple code and describes problems caused by language interaction. Our solution is described in Section 3 and the implementation is outlined in Section 4. The Section 5 discusses, how our approach solves the problems from the Section 2 and what are the limitations of our approach.

2 Problem

2.1 Example

Consider code in Figure 1 and Figure 2 that demonstrates interaction of Java and Smalltalk languages. In Figure 1, there is a method `sayHello` that selects language according to the locale and print appropriate greeting. The `sayHello` method expects a map with translations to be passed as a parameter. In Figure 2, there is a Smalltalk code that prints greeting using the Java code shown in Figure 1. The Smalltalk creates a translations as an instance of class `Dictionary` and then invokes `sayHello` method to print the greeting.

During the invocation of `sayHello` method on an instance of `Multilanguage-HelloWorld` class, an instance of Smalltalk `Dictionary` is passed to Java method that expects an instance of `java.lang.Map`. In that case, we say that the Smalltalk object *crossed the language boundary*.

2.2 Problem description

The problem is that the `Dictionary` cannot be used as parameter of `sayHello` method directly – it is a different object from completely different type hierarchy with different set of methods. Yet we intuitively feel, that the `Dictionary` is Smalltalk equivalent of `java.util.Map`. Both are used to store values under arbitrary key.

```
public class MultilanguageHelloWorld
{
    public void sayHello(HashMap dictionary)
    {
        String key = getLocale().getLanguage();
        System.out.println(dictionary.get(key));
    }
}
```

Fig. 1. Java class `MultilanguageHelloWorld` that can print greetings according to the locale.

```
greetings := Dictionary new
    at: 'en' put: 'Hello World';
    at: 'cs' put: 'Ahoj světe'.

MultilanguageHelloWorld new
        sayHello(greetings).
```

Fig. 2. Smalltalk code interacting with Java object - `MultilanguageHelloWorld`

If we want to let Java and Smalltalk code from Figure 1 and Figure 2 interact smoothly, there are two basic approaches; First, do not create an instance of `Dictionary` – create an instance of `java.lang.HashMap` class from the very beginning. Or second, if necessary, create a new `HashMap` object and copy data from the `Dictionary` to the `HashMap`.

In the first case, there may arise problem when the object creation is not under our control. For example, if the translation mapping is obtained from a third-party library which cannot be modified. The second approach is time consuming and has higher memory requirements. We have to take care about the object identity as well: if we created a new object every time the object is passed from Smalltalk to Java, multiple Java `HashMaps` would represent the same Smalltalk `Dictionary`. Moreover, data should be kept in sync: if something changed in the Java `HashMap`, we should update the Smalltalk `Dictionary` object.

Usage of a proxy [6] object is third, more advanced approach. The proxy eliminates problems with data synchronization. Nevertheless problems with identity remains and we have to map proxies to their subjects which causes extra performance and memory overhead.

3 Our solution

As mentioned before, our approach represents single object in various languages by the only one physical object with dynamically changed behaviour. Object behaviour in specific language is described by a structure that we call *behaviour object*. In other words, the *behaviour object* describes behaviour of given object in scope of given language. In most of languages, the behaviour object is its class, in prototype languages the behavior object might be represented by object map [3]. Any physical object may be associated with as many behaviour objects as is the number of languages in which the object is used. Whenever an object crosses language boundary we dynamically change a behaviour object according to the actual language. The `greetings` object from Figure 2 would have two behaviour objects associated – one for Smalltalk language representing the `Dictionary` class and one used in Java representing the `java.util.Map`.

Next important part of our approach is a mapping of an object state. We will call an ordered set of object fields as *an object layout*. *A primary object layout* is then an object layout defined by the language where the object was instantiated. We will call a set of object fields and their respective values as *an object state* – an object state is an object layout with values. Similarly, *a primary object state* is a state of the object with primary layout. Any method in any language may change an object state. Unfortunately, each behaviour object may require different object layout. Because we share the same physical object among languages, a mapping function has to map primary object state to desired object state and vice versa.

The idea of shared behaviour and mapped state is depicted in Figure 3. In the upper left-hand corner there is a physical object `java.lang.String` composed of behaviour and state. In the upper right-hand corner there is a similar structure for Smalltalk `String`. In the bottom, there is a composed object – one physical object with both, Smalltalk and Java behaviour. The behaviour is simply added, the state has to be mapped from Smalltalk to Java.

We will describe our approach more formally now. Let's have a virtual machine VM which is able to interpret native language L_1 and alien language L_2. Let's have a program P_1 written in L_1 and a program P_2 written in L_2. P_2 interacts with P_1. As an input parameter, P_1 expects an object O_1 with behaviour described by a behaviour object B_1. P_2 creates an object O_2 with a primary layout $A_2 = f_{21}, f_{22}, \dots, f_{2n}$ and with behaviour described by a behaviour object B_2. The object layout A_2 with values is an object state S_2. B_2 differs from B_1. B_1 expects an object to have a layout $A_1 = f_{11}, f_{12}, \dots, f_{1m}$. The object layout A_1 with values is an object state S_1. We want to use O_2 in P_1 as O_1. An example could be find in Figure 1, Figure 2 and described in Section 2.1.

We need to define mapping from S_2 to S_1. Such mapping has to satisfy two requirements. First, an appropriate value has to be determined from S_2 when the value of a field $f \in A_1$ is needed. Second, S_2 has to be updated accordingly when a field $f \in A_1$ is being set. If the layouts of A_1 and A_2 are identical, the mapping is trivially identity mapping. If it is not possible to map S_2 to S_1, O_1 and O_2 could not be considered to be equivalent in L_1 and L_2 – they have too little in common. In the rest of cases, the mapping has to be specified explicitly.

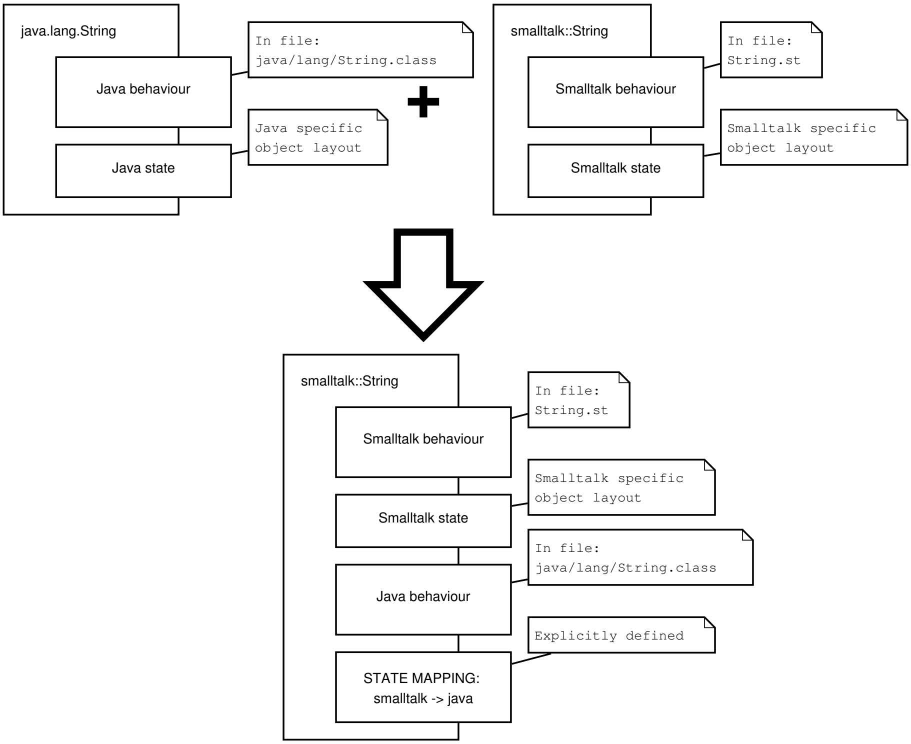


Fig. 3. One physical object with multiple behaviours (in the bottom) is (in the top). The behaviour is added, the state is mapped.

It is also necessary to provide mapping that maps languages and behaviour object, i.e., that the object O_1 with behaviour of B_1 in language L_1 will be associated with behaviour B_2 in language L_2 and vice versa. Whenever a message is sent to O_2 from L_1 (L_2 respectively), a message selector will be looked up in B_1 (B_2 respectively). During a program execution, various situations may occur:

- When a method is called on O_2 from P_1, the method is looked up in B_1.
- When a field $f \in A_1$ of O_1 is being read from P_1 and A_2 is the primary object layout, then the mapping from S_1 to S_2 is used to compute the value of f based on S_2.
- When a field $f \in A_1$ of O_1 is being set from P_1 and A_2 is the primary object layout, the mapping from S_1 to S_2 is used and the S_2 is updated.
- When the object O_2 is passed from P_2 passed to P_1 (O_2 crosses the language boundary), B_1 is assigned to O_2.
- When the object O_2 is passed from P_1 back to P_2, B_2 is assigned to O_2 again.

4 Implementation

We have validated our solution on Java and Smalltalk programming languages. We use Smalltalk/X virtual machine to interpret Java and Smalltalk language. We employ metaobject protocol [11] that is implemented in Smalltalk/X VM [14] to change method and field lookup semantics.

We use standard Smalltalk class as a behaviour object for Smalltalk objects. We use special Smalltalk object similar to Java class as a behaviour object for Java objects. Essentially, some objects have two classes – one for Smalltalk and second for Java. We modified method lookup in order to reflect an existence of multiple behaviour objects per object as follows:

```
Lookup>>lookupMethodForSelector:selector
        for:receiver
        withArguments:argArrayOrNil
        context: context
    | behaviour |
    behaviour := receiver behaviourObjectFor: context language.
    behaviour lookupMethodForSelector: selector
              withArgumets: argArrayOrNil.
!
Object>>behaviourObjectFor: language
    ˆ ObjectRegister instance
        getCorrespondingClassOf: self primaryBehaviourObject
        inLanguage: language.
!
Object>>primaryBehaviourObject
    ˆ self class
!
```

Lookup object, which is responsible for lookup of appropriate method for foursome `selector, receiver, argument array, context` and which is called before each message send, delegates the lookup to the behaviour object. Behaviour object knows appropriate method lookup algorithm and which methods are available in current context. Behaviour object depends on actual language.

Furthermore we modified field accessor functions to be able to apply mapping between different states as follows:

```
Lookup>>getFieldForFieldName:fieldName
        for:receiver
        context: context
    | behaviour primaryBehaviour |
    behaviour := receiver behaviourObjectFor: context language.
    primaryBehaviour := receiver primaryBehaviourObject.
    ˆ StateMapping instance
        getFieldNamed: fieldName
        fromBehaviour: behaviour
        toBehaviour: primaryBehaviour
        forObject: receiver.
!
```

Lookup object, which is responsible for accessing instance variables and which is called before each field access, delegates execution to the mapping object, which will determine particular field value from primary object state.

Last but not least, we introduced global map, where the equivalent types may be registered together with the state mapping functions as follows:

```
ObjectRegister>>addBehaviour: behavirouObject
                to: primaryBehaviourObject
    | behaviourObjectCollection |
    behaviourObjectCollection := self at: primaryBehaviourObject.
    behaviourObjectCollection add: behavirouObject.
!
```

A demonstration of our solution's abilities is depicted in Figure 4 and Figure 5. Equivalent codes and outputs in other languages will be described later in Section 6 which compares our implementation with another ones.

```
SOURCE:                          OUTPUT:
string := 'Smalltalk string'.
smalltalkInfo info: string.      info from Smalltalk world
    // string class                  class: String class
    // string hash                   hash: 197479768
javaInfo info: string.           info from Java world
    // string.getClass()             class: java.lang.String
    // string.hashCode()             hash: 7110656
java equals: string and: string  object equals:
    // string1 == string2            true
```

Fig. 4. An interaction of Smalltalk String with Java code.

In the Figure 4 there is a code which sends Smalltalk string to (i) Smalltalk object, (ii) to Java object and (iii) compares the same instances of the string in Java environment. The figure is divided into two parts. There is a source in the left and output in the right. The `smalltalkInfo`'s method `info` prints a class and hash code of a parameter. It demonstrates how does object look like in Smalltalk context. The `javaInfo` variable is pointer to the class written in Java and compiled to the Java bytecode. The `javaInfo`'s method `info` prints a class and hash code of a parameter as well. It demonstrates how does object look like in Java context. The `javaInfo`'s method `equals` compares identity of parameters and prints `true` if objects are identical, `false` otherwise. It demonstrates that the same object has the same identity in alien language. As you can see, the String object has appropriate class and hash in both of the languages.

The Figure 5 is divided into two parts as well – source in the left and output in the right. The `smalltalkInfo`'s method `info` is the same as in Figure 4. It prints class and hash code of a parameter. The `javaInfo`'s method `info(Object o)` prints a

```
SOURCE:                          OUTPUT:
set := HashSet new with: 1
                   with: 6.
smalltalkInfo info: set.         info from Smalltalk world
                                     class: HashSet class
// info(Object o)                    hash: 6537216
javaInfo info: set.              info(Object o) from Java world
                                     class: java.util.HashSet
//info(Set s)                        hash: 6537216
javaInfo infoSet: set.           infoSet(Set s) from Java world
                                     class: java.util.HashSet
//info(Set s)                        hash: 6537216
javaInfo infoHashSet: set.       infoHashSet(HashSet s) from ...
                                     class: java.util.HashSet
                                     hash: 6537216
```

Fig. 5. Intraction of Smalltalk object with Java code.

class and hash code of a parameter – it demonstrates that Smalltalk object may be handled as Java object even though `java.lang.Object` is not anywhere in Smalltalk class hierarchy. The `javaInfo`'s method `info(Set s)` demonstrates that Smalltalk object may be handled as Java interface. The `javaInfo`'s method `info(HashSet s)` demonstrates that Smalltalk object may be handled as ordinary Java class.

5 Discussion

In case an object is shared between multiple languages and its behaviour is dynamically changed according to the actual language, following problems are naturally solved:

Object identity The object identity is based on an object pointer comparison. Since we represent objects by the same pointer in computer memory, no problem arises.

Explicit copy If there is no support for automatic object conversions between, programmers have to take extra care while passing object across the language boundary. It may happen that an alien object with inappropriate behaviour will be used that may rise an exception. The error may be prevented by explicit call of a conversion method. On the other hand, if the behaviour is changed automatically, the work with alien objects is transparent – they look like native objects. No extra care has to be taken while passing object across the language boundary.

Data synchronization If objects has to be copied while crossing the language boundary, synchronization of data has to be ensured. Our solution work with the same data so it is not a deal any more.

Memory overhead Object copy implies memory overhead since all data are duplicated. Proxy objects may be light-weighted as to not consume too much memory, nevertheless due to necessity to preserve an identity, an extra memory is consumed by (global) mappings of objects to their respective proxies. Such a mapping is not only memory consuming but also requires proxies to be weak-referenced. Weak references affects garbage collector performance since all weak references must be treated specially. In our solution, objects are shared between multiple languages and so the memory is not occupied redundantly. Behaviour objects does not cause any memory overhead as well, since they are already present in particular languages.

Questions regarding the reflective facilities may arise.

Object class Object class could be obtained by sending appropriate method (`class` in Smalltalk, `getClass()` in Java). The return value is metaobject which keeps information about methods, fields, subclasses, super class and others. It could be said that the return value is the behaviour object (in some form) currently associated with the given object. Our technique does not affect this functionality. For each language, appropriate object representing the class is returned. From the point of any particular language, an object has one class.

Object superclass Object superclass is stored in its behaviour object. Since the correct behaviour object is always returned, asking it for a superclass will return a corresponding superclass in scope of given language.

Super sends Since the problems has not arose in previous case, it is not problem to invoke super send. Nevertheless if Y_2 `extends` X_2 in language L_2 (with object layouts A_{Y2} and A_{X2}) and Y_1 exists in language L_1 (with object layout A_{Y1}) and Y_1 is used in language L_2 as Y_2, the Y_1 must provide mapping from A_{B1} to $A_{B2} \cup A_{A2}$. In other words, Y_1 must provide mapping to the complete object layout of Y_2 – including superclasses.

5.1 Implementation limitations

There are several possible implementations of our approach. We have chosen to profit from metaobject protocol implemented in Smalltalk/X as described in Section 4. Another suitable metaobject protocol is provided by Dynamic Language Runtime [5] framework built on top of Common Language Runtime [13]. Unfortunately, it is not possible to integrate C# and IronRuby [2] or IronPython [1] this way, because existing C# does not use Dynamic Language Runtime.

In Smalltalk, another techniques like `doesNotUnderstand:` hook and Java bytecode instrumentation could be used. The `doesNotUnderstand:` hook allows method lookup customization, but this technique negatively influences performance. The bytecode instrumentation may be used to replace method call in bytecode by another routine in bytecode that takes multiple behaviour objects into the account. The get field and set field bytecode instructions may be replaced by similar routine that take state mapping into the account.

6 Related Work

6.1 JRuby

JRuby [4] is an implementation of Ruby language running on top of Java Virtual Machine. Generally, JRuby objects may interact with Java code. Nevertheless there are some “pain points”. In case of Strings, they may be shared between JRuby and Java without any limitations. A class of a String object changes appropriately, a hash code is computed correctly and an identity is preserved. This demonstrates code depicted in Figure 6.

```
SOURCE:                         OUTPUT:
string = "ruby string"
rubyInfo.info(string)           info from Ruby world
    // string.class                 class: String
    // string.hash                  hash: 250737224
javaInfo.info(string)           info from Java world
    // string.getClass()            class: java.lang.String
    // string.hashCode()            hash: 916834583
javaInfo.equals(string, string) object equals:
    // string1 == string2           true
```

Fig. 6. Interaction of Ruby String with Java code.

The code in Figure 6 is written in Ruby which interacts with Java. The code is equivalent to the code in Figure 4 which is written in Smalltalk. Regarding strings, there is no difference between abilities of JRuby and our solution.

Generally, JRuby objects may be used as a parameter whenever the parameter is `java.lang.Object` because JRuby objects inherit from `java.lang.Object`. Moreover, JRuby object may be used as a parameter of Java method in case the parameter is Java interface and the Ruby object implements the interface. Yet, if a Java method expects standard object (subtype of `java.lang.Object`), exception is raised. This demonstrates a code depicted in Figure 7.

The code in Figure 7 is written in Ruby which interacts with Java. The code is equivalent to the code in Figure 5 which is written in Smalltalk. Source is in the left, output is in the right. As you can see, JRuby allows to pass Ruby object to methods, which expects `java.lang.Object` and Java interface, but not `HashSet`. Our implementation allows to pass Smalltalk object to any of the methods.

6.2 Jython

Jython [9] is an implementation of Python language running on top of Java Virtual Machine. Jython objects may interact with Java code, but there are some “pain points” as well. In case of Strings, they may be shared between Jython and Java without limitations. A class of an object is changed appropriately, a hash code is computed correctly and an identity is preserved. This demonstrates code depicted in the Figure 8.

There is a code written in Jython which interacts with Java objects in the Figure 8. The code is code equivalent to the code in Figure 4 which is written in Smalltalk. Source is in the left, output is in the right. Again, regarding strings, there is no difference between abilities of Jython, JRuby and our solution.

```
set = Set[1, 3, 4, 11]
rubyInfo.info(set)
                                 info from ruby world
// info(Object o)                    class: Set
javaInfo.info(set)                   hash: 24118174
                                 info(Object o)
// infoSet(Set s)                    class: org.jruby.RubyObject
javaInfo.infoSet(set)                hash: 24118174
                                 infoSet(Set s)
// infoHashSet(HashSet hs)           class: ....InterfaceImpl
javaInfo.infoHashSet(set)            hash: 21279119
                                 infoHashSet(HashSet hs)
                                     cannot convert class
                                     org.jruby.RubyObject to
                                     java.util.HashSet
```

Fig. 7. Interaction of Ruby object with Java code.

```
SOURCE:                          OUTPUT:
string = "jython string"
jythonInfo.info(string)          info from Jython world
    // string.__class__              class: <type ’str’>
    // string.__hash__()             hash: 1857618127
javaInfo.info(string)            info from java world
    // string.getClass()             class: java.lang.String
    // string.hashCode()             hash: 1857618127
javaInfo.equals(string, string)  object equals:
    // string1 == string2            true
```

Fig. 8. Interaction of Jython String with Java code.

Yet, it is not easy to use Jython object as parameter of Java method. There is a mechanism called *Object Factory* in the Jythonbook [8] but it requires lots of code overhead. The mechanism cannot be used in all use cases. Generally, it is not possible to pass Jython's `ImmutableSet` instance into the Java method expecting either `java.util.Set` or `java.util.HashSet`. This is a difference between our solution and Jython, since our solution is not limited in these use cases.

7 Conclusion

In this paper we have presented a dynamic behaviour switching mechanism to support language interoperability. When an object is passed from one programming language to another, its behaviour is dynamically switched to what the other language expects, allowing programmers to work with alien objects in a natural way. The same physical object is used in all languages, therefore there is no runtime overhead caused by copying objects and by maintaining object identity. A mapping from class in one language to corresponding class in the other language is provided by user as well as a mapping of object state.

References

1. IronPython, August 2010.
   `http://ironpython.net/.`
2. IronRuby, August 2010.
   `http://ironruby.net/.`
3. Craig Chambers, David Ungar, and Elgin Lee. An efficient implementation of SELF — a dynamically-typed object-oriented language based on prototypes. In *Proceedings OOPSLA '89, ACM SIGPLAN Notices*, volume 24, pages 49–70, October 1989.
4. Charles Nutter et. al. JRuby Project, August 2010.
   `http://jruby.org/.`
5. Bill Chiles and Alex Turner. Dynamic Language Runtime, August 2010.
   `http://dlr.codeplex.com/wikipage?title=Docs%20and%20specs.`
6. Erich Gamma, Richard Helm, John Vlissides, and Ralph E. Johnson. Design patterns: Abstraction and reuse of object-oriented design. In Oscar Nierstrasz, editor, *Proceedings ECOOP '93*, volume 707 of *LNCS*, pages 406–431, Kaiserslautern, Germany, July 1993. Springer-Verlag.
7. Kris Gybels, Roel Wuyts, Stéphane Ducasse, and Maja D'Hondt. Inter-language reflection – a conceptual model and its implementation. *Journal of Computer Languages, Systems and Structures*, 32(2-3):109–124, July 2006.
8. Josh Juneau, Jim Baker, Victor Ng, Leo Soto, and Frank Wierzbicki. Jython Book v1.0 documentation, March 2010.
   `http://www.jython.org/jythonbook/en/1.0/.`
9. Jython, February 2011.
   `www.jython.org.`
10. Jevgeni Kabanov and Rein Raudjärv. Embedded typesafe domain specific languages for Java. In *PPPJ'08: Proceedings of the 6th International Symposium on Principles and Practice of Programming in Java*, pages 189–197, Modena, Italy, 2008. ACM.
11. Gregor Kiczales, Jim des Rivières, and Daniel G. Bobrow. *The Art of the Metaobject Protocol.* MIT Press, 1991.
12. Jacob Matthews and Robert Bruce Findler. Operational semantics for multi-language programs. *SIGPLAN Not.*, 42(1):3–10, 2007.
13. E. Meijer and J. Gough. Technical overview of the common language runtime, 2000.
14. Jan Vraný. Supporting multiple languages in virtual machines. Dissertation thesis, Czech Technical University, December 2010.

Task Scheduling in Data Stream Processing*

Zbyněk Falt and Jakub Yaghob

Department of Software Engineering, Charles University,
Malostranské nám. 25, 118 00 Prague, Czech republic
{falt, yaghob}@ksi.mff.cuni.cz
http://www.ksi.mff.cuni.cz/

Abstract. One possible technique of data processing is its transformation into a data stream and the execution of particular operations on the data tuples. These operations can be usually processed concurrently especially when the plan of operations is branched. Therefore, this way of data processing is suitable for evaluation in parallel environment. On the other hand, the ordering of the execution of tasks associated with the operations is closely related to the utilization of the hardware components. For that reason, the task scheduler is very important in these systems. In this paper, we introduce our implementation of a task scheduler which deals well with various hardware factors such as caches and NUMA[1] factor.

1 Introduction

As the amount of data which are intended for processing becomes larger, new approaches for their processing are researched. Since the performance of the systems has been recently increased mainly through the addition of computational units, the parallel processing of data is a natural evolution in this research area. New algorithms, new approaches, and new technologies, which utilize this direction of progress, are developed.

On the other hand, the development of parallel data processing is more difficult than single threaded development. The developer must solve issues such as the decomposition of the problem to independent parts which may be processed in parallel, thread management, their synchronization and communication.

Of course, there exist many frameworks which try to make these things easier. One of the approaches, that makes especially the processing of data queries easier, is the transformation of input data into data streams and the execution of particular operations on that streams. The execution plan looks like directed graph where nodes are the operations and edges determine dataflow among the operations. The biggest advantage of this approach is that developer of this plan does not have to take parallelization into account since this is done automatically

* This paper was supported by the grant SVV-2011-263312 and by the grant agency of Charles University (GAUK), project no. 277911.

[1] Non-Uniform Memory Access

during the evaluation. It is possible, because operations on different parts of the streams or whole branches of the plan may be processed concurrently.

One of the systems that implements this idea is Bobox [6]. The main aim of the Bobox project is to process semantic and semistructured data effectively [7]. Currently, support for SPARQL [15] query language is under development and partial support for XQuery [8] and Tri Query [5] language is already developed.

The contribution of this paper is a presentation of the scalable scheduling techniques for systems which process data streams. These techniques deal well with different hardware factors such as size of caches and NUMA factor. Therefore they enable to evaluate data queries faster on modern systems. Although they are analyzed and tested on the Bobox system, they can be easily adopted by other similar systems.

The remainder of this paper is as follows. In the Section 3.1, there is the description of execution plan evaluation and tasks creation. In the Section 4, we measure and analyze the influence of various hardware factors on the efficiency of the system. The Section 5, contains detailed description of the implementation of our scheduler and in the Section 6, we present and analyze an experimental comparison between our old simple and new implementation of the scheduler. The Section 7 summarizes the results and introduces our future plans.

2 Related work

Parallel processing of data stream is a very actual research topic and many systems similar to the Bobox are being developed, for example Borealis [1], STREAM [2], Nile [12] or NiagarsST [14]. However, the aim of all the systems is mainly the processing of an infinite data streams or real-time data streams. This is one of the biggest differences between them and the Bobox, because the main scope of the Bobox is efficient processing of the offline finite data streams.

Of course, these differences determine different requirements on the scheduler and the scheduling strategies. The common requirements for all systems are throughput and efficient utilization of available resources. Additionally, in the processing of infinite or real-time data streams, memory usage and response time are also very important and some papers address this problems [13], [3] or [17].

In the Bobox-like systems the factors of response time do not have to be taken into account, as they are not meant to be the frameworks for processing real-time data. The finiteness of the input data ensures that the memory usage will not grow beyond control.

The other major difference is that the Bobox supposes there is no parallelism in the operator evaluation. This means that in contrary to other systems each operator is allowed to run only on at most one thread. This restriction makes operators implementation easier. On the other hand, the only way of achieving concurrency in the operator implementation is their decomposition to several atomic operations. For example, sorting can be decomposed to several single threaded sorting operators, which might be evaluated in parallel. Their results can be then merged together by the net of merge operators.

Because one operation can be composed of many partial operations, the scheduler must be optimized according to this fact. The main reason is that the communication among these suboperations is more intensive than among the operators. To deal with it, our scheduler optimizes the work with the CPU cache and tries to keep this communication in the caches as much as possible.

The strategy implemented in the paper [10] also optimizes the utilization of the CPU caches. However, in contrast to our work, it tries to optimize the accesses to caches during operator evaluation. We focused on the utilization of the caches to speed up the communication among operators.

Problems of task scheduling are also a very important part of frameworks or environment which are not related to data streams processing, as shown for instance in TBB [16] or OpenMP [9], [11]. As these papers take into account cache related problems and solve them via increasing the data locality, they are not concerned with factors such as NUMA factor.

3 Evaluation of an execution plan

As mentioned briefly in Section 1, the execution plan looks like a directed graph. The nodes of this graph represent operators and the edges determine the dataflow in the plan. Operators can have an arbitrary number of input edges through which they receive data tuples, and an arbitrary number of output edges along which they send the resulting tuples. Since tuples are typically small and some overhead is related to their transfer, they are grouped together into packets. This grouping helps to reduce the communication overhead. The other important thing is, that the operators are allowed to communicate with each other only through these packets. Thus, they should not share any variables or other resources.

We denote executions plans as requests in the rest of the paper, because each plan is typically related to some data query request.

3.1 Tasks creation and their types

The fact that each operator can run on at most one thread determines the set of states of each operator. These states are:

- *Nothing* – The operator has nothing to do.
- *Scheduled* – The operator has been scheduled to perform its operation.
- *Running* – The operator is processing input data or performing another operation. After that it will be switched to the *Nothing* state.
- *Running and scheduled* – The operator has been scheduled during the *Running* state. When the operator finishes its operation it will be switched to the state *Scheduled*. The scheduling requests are serialized in this way and therefore the operator cannot be run twice at a time even if it is scheduled multiple times.

When a packet is received by the operator, it is switched to state *Scheduled* or *Running scheduled*. Every time the operator is switched to the state *Scheduled*, new task is created. The objective of the scheduler is to select tasks and run them on the threads which it has chosen.

The receipt of the packet is not the only event when the task is created. The other situation is, for example, when the operator which generates new data sends data packet, but it still has many other packets remaining. In this case, the operator schedule itself again after the packet is sent and the next time it is run, it sends the next packet etc.

It is obvious that the tasks associated with packet reception require a bit different handling. The fact that the packet was received probably means that the content of the packet is hot in cache. Thus, the earlier the associated task will be performed, the faster this performance probably will be. Other types of tasks are not directly associated with an packet, therefore it does not matter when or even where these tasks will be run. The first task type is called immediate and the second task type is called deferred.

4 Factors influencing efficiency

This section contains a list of the most important hardware factors which the scheduler should take care of. The importance of these factors is experimentally verified on the Bobox system.

4.1 Cache

One of the goals we wanted to reach was an effective utilization of caches. One way of doing it is to try to keep the content of packets in the cache. Therefore, the packet transfer does not cause many cache misses. If we consider only this factor, we conclude that the smaller packets the better the performance is since they fit better into the cache. On the other side, there is some overhead with this transfer such as synchronization of data structures, scheduler decision algorithms etc. If we consider only this overhead, then bigger packets decrease this overhead and increase the efficiency.

Our task was to find a reasonable compromise between these two extremes. We measured the relation between the size of packets and the efficiency of the system. We executed the same request with different packet sizes. The results are shown in Figure 1 and confirm the hypothesis. With smaller packets the overhead grows up and with larger packets there is an overhead with cache misses.

According to the plot, the ideal size of packets is slightly lower than the size of L2 cache, which is 256KB (see Section 6 for more detailed hardware specification). It can be easily explained since not only the packets are stored in the cache, but there are also data structures of scheduler, memory allocator or operators. These data structures are heavily accessed during the evaluation, so keeping them also in the cache improves efficiency. Therefore the packet size must be lowered by the size of these structures in order that they all fit into the cache. Unfortunately, the size of these structures is hard to estimate.

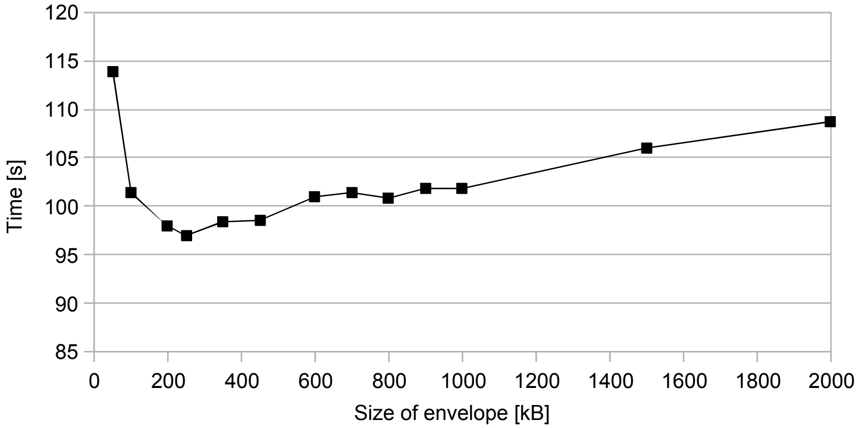


Fig. 1. The influence of the packet size on the system performance

4.2 NUMA factor

The number of processors in a SMP system cannot grow infinitely. As the number of processors in the system is increasing, shared resources are turning into bottlenecks. The most critical shared resource is the main memory. Therefore NUMA systems are being developed as a solution to this problem.

Basically, the NUMA system consists of several nodes. Each node acts as SMP system, i.e. it has its own local main memory. These nodes are connected together and the whole system shares the physical address space. Thus, one node can access local memory of another node, but this access is slower than an access to its local memory. This slow down is known as NUMA factor. The NUMA factor tells us how many times slower is access to non-local memory in the comparison to access to local memory.

It is obvious that processor should access preferably its local memory. In the opposite case, the computation is slowed down by the NUMA factor. Of course, this problem is related more to memory allocation than to scheduling, but NUMA non-aware scheduler can cause performance penalties, too. For example when the scheduler move a computation, which has allocated some objects in its local memory, from one node to another.

To measure the influence of the NUMA factor on the Bobox system, we did the following test. Firstly we performed a request on one NUMA node and all allocations return its local memory. Secondly, we performed the same request on the same NUMA node, but all the memory was allocated on another node. The results (see 2) proves that the influence of NUMA factor cannot be ignored.

For a comparison, we performed synthetic test which measures the NUMA factor of our testing system. The test measured time needed for reversion of the order of items in large array of integers. The test were run for all combinations of nodes where the algorithm was performed and nodes where the memory was

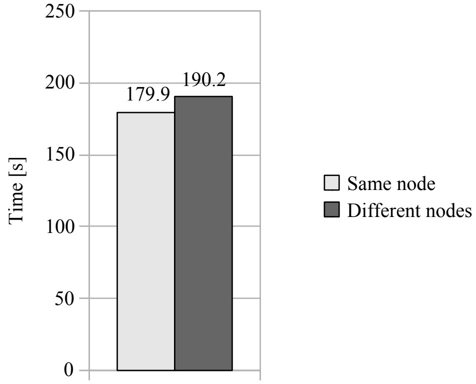


Fig. 2. The influence of the NUMA factor on the system performance

allocated. The results are shown in the Table 1, where all numbers are recalculated relatively to the lowest times (which are naturally on the diagonal). Nodes are numbered from 0 to 3. The factor is quite low in the system, so the difference between both methods in the Figure 2 is not so significant, but the higher the factor is, the bigger difference will be.

	0	1	2	3
0	1.0	1.4	1.5	1.4
1	1.5	1.0	1.4	1.4
2	1.4	1.5	1.0	1.5
3	1.5	1.5	1.5	1.0

Table 1. Matrix of NUMA factors in the system

5 Scheduler

5.1 Data structures and management of tasks

Data structures of the scheduler reflect the hardware structure of the host system and the fact that system may be evaluating many requests at a time. Therefore each NUMA node contains the list of all requests it is currently evaluating, each request contains the list of all deferred tasks and each thread contains the list of immediate tasks which were created by this thread. When the host system is not NUMA system but ordinary SMP, it is still considered a NUMA system with only one node.

The algorithm of what to do when a new task is created is quite simple, because the real work is done by the second part of the algorithm, which is described below. A new immediate task is inserted into the list of immediate tasks of the thread which created the task. A new deferred task is inserted into the list of deferred tasks of the request to which the task belongs.

When a worker thread finishes its current task, it starts to find another task which it will execute. The algorithm which selects this task is a bit more sophisticated and its goal is to maximize the usage of the principle of locality, which increases the efficiency of the system. Our strategy is to find the first task in this order:

1. The newest task in the list of immediate tasks that belongs to the thread. Since this task is immediate, it is associated with some data packet which was recently created. The content of the packet was probably in the cache after the creation. The fact that the task is the newest increases the probability that no other packet moves the packet out of the cache.
2. The oldest task in the list of deferred tasks that belongs to request, that was evaluated by the thread for the last time. Each request has probably some data loaded in the cache and switching to another request would cause more cache misses than the continuation with the same request. There are two reasons, why we decided to choose the oldest task from the list.
   The first is, that if we chose the newest task, then the execution plan would be evaluated in the DFS[2] manner instead of BFS[3]. But BFS manner tends to generate more concurrent tasks than the DFS, so the level of parallelism is higher.
   The other reason is, that each operator has its own input buffers for incoming packets and deferred tasks are typically used for deferred packet creation (see Section 3). Thus, the later this task will be executed, the more probable is, that the input buffer of the successive operator will be empty. This increases probability, that the newly created packet will not be rejected by the operator and also probability, that the oldest packets in the input buffer (i.e. packets which the successive operator will process) are hot in cache.
3. The oldest existing task of the oldest request which the current node is evaluating. Therefore old requests are processed primarily. Each request allocates many resources during its evaluation – memory, files etc. These resources are usually allocated successively during the request evaluation. It is obvious that if all request were processed with the same priority, then the total amount of allocated resources would be increasing faster than if one request is processed preferably and others are suppressed. Additionally, when the oldest request is finished, its allocated resources will be freed and will be available for other requests. Therefore, this strategy tries to minimize the number of allocated resources during requests evaluation.
4. The oldest task in the list of immediate tasks of another thread from the same node. This stealing of the task violates the principle of locality, on the other hand it is better to utilize idle thread than let it sleep.
5. If no such task exists, then the seeking thread is suspended.

[2] Depth-first search

[3] Breadth-first search

This architecture causes, that one request can be evaluated only on one node at a time. This is a suitable behavior for NUMA systems because during the evaluation threads work only with its local memory. On the other hand, this might be a problem for request with high degree of parallelism. In that case, only a part of the system performance is used for its evaluation. Another problem is that system might become unbalanced, which means, that some nodes are overloaded, while others are idle.

The second problem is partially solved by the algorithm, which chooses a node that has the most free resources available when a new request is created and passes the request to that node. The algorithm assumes free resources to be the free memory and the number of idle threads. This solution is partial because when the request is created, it is unknown how many resources it will be spending. Therefore the initial decision may be wrong in the global context. This is solved by a dynamic load balancing described below.

The first mentioned problem is solved by the load balancing as well.

5.2 Other algorithms

Load balancing When there is at least one idle logical processor in the system, a special load balancing thread is running. This thread evaluates every 100ms load of the system. When there is at least one overloaded and at least one partially idle node, then load balancing algorithm tries to solve this imbalance. The algorithm selects the newest request of the overloaded node and moves it to the idle node. The idea is that the newest request has probably the smallest amount of allocated memory, therefore this transfer should be the most painless.

If there is no request to move (i.e. node is overloaded and is processing only one request), then the request becomes shared with the idle node. At this moment, the request starts to be evaluated by both nodes. Of course, it is possible that one queue is shared among more than two nodes. This happens when the request has a high level of parallelism and is able to overload more nodes.

Sleeping threads A very important attribute of task scheduler is whether it can put to sleep the threads which have nothing to do. An active waiting for a next task is really inefficient. There are three main reasons. The first one is that an active waiting keeps the CPU units busy. For cores with Hyper-Threading technology, where these units are shared between two logical processors, this behavior is very undesirable since the waiting threads slow down the other threads.

The second reason is, that an active waiting means a repeated checking whether there is a new task. Because of the synchronization of the shared structures this checking loads the bus, memory or cache coherency protocol. Thus, this approach decreases performance even of non-waiting threads.

And the last one is, that almost all modern systems have some kind of power saving technology which switches idle processors to low-power state. Therefore, thread sleeping reduces the power consumption of the host system.

However, the thread sleeping brings one problem we have to cope with. When the scheduler allows thread sleeping, there must be a mechanism for deadlock prevention. This is related to the situation when there are more tasks than running threads and though idle threads exist. This situation must be avoided.

Our solution is quite simple. Each NUMA node keeps the number of the tasks for which it is responsible, i.e. a node must ensure that all these tasks will be processed. The thread must not be suspended when the number of tasks is greater than the number of running tasks. Therefore all tasks for which the node is responsible will be processed before all threads will be put to sleep. When any task or request is created and there is an idle thread in the corresponding node, then this thread is woken up.

The implementation must also prevent the situation when some request is shared but some nodes are idle because they are not responsible for any task of this request. We have decided for a simple but effective solution – the responsibilities for new deferred tasks of the request are distributed in the round-robin way among all sharing nodes.

The implementation of thread sleeping itself can also influence the performance of the system. The straightforward solution is to created semaphore for each node. The value of the semaphore would be equal to the number of tasks for which is the node responsible. New task would increase the value and each thread would decrease the value before it would try to find another task.

Unfortunately each change of the semaphore value yields to a system call, which can be quite slow. Our implementation tries to avoid this. Each thread has its own binary semaphore. Moreover the number of running threads and the number of tasks are counted in an atomic variable. Before a thread is suspended, the number of running thread is increased and the semaphore of the thread is acquired. When the number of running threads is lower than the number of tasks, then the number of running threads is increased and one semaphore of some selected thread is released.

This implementation ensures, that when the node is fully loaded, i.e. all threads are running, then no operation is called on semaphores. Furthermore there is no global semaphore which could become a bottleneck.

NUMA support NUMA system support is already integrated in the architecture of the scheduler. Each node has its own task queue and the scheduler tries to keep all the requests only on one node. Only in the case when one node is overloaded it tries to share computations among more than one node. This strategy prevents it from redundant movement of requests among the nodes.

The next advantage follows from the fact that only the queues of deferred tasks are shared. Deferred tasks are not related so tightly to data flow as the immediate tasks. If an packet is created (and its memory allocated) and sent to other operator, the task associated with this packet will be processed on the same node (and the thread will work with its local memory). The resulting packet of the operation will be processed on the same node as well for the same reason.

On the other hand, access to non-local memory cannot be fully avoided, because operators such as for example merge or join can have more inputs which are processed together.

CPU cache support It follows from the measurement in the Section 4 that performance of the system depends on the size of the packets and the size of the CPU caches. The optimal size according to the results is the same as size of L2 cache lowered about the size of data structures of scheduler, allocator etc.

The problem is that this size is hard to estimate, since it depends on many factors, such as behavior of the operators or the structure of the input data. We have decided to solve this problem in a very simple way. As the optimal size of an packet, we consider one half of the L2 cache size. For this size, all experiments gave the best results. But we plan to solve this problem better in future (see Section 7).

6 Results

To measure the benefits of the new scheduler, we used system with 4 Intel Xeon E7540 running at 2GHz CPUs. Each processor has 6 cores with the Hyper-Threading support. Altogether the whole system has 48 logical processors. Each core has its own 32kB data and 32kB instruction L1 cache and 256kB L2 cache. All cores in the one processor share the 18MB L3 cache. The system has 4 NUMA nodes; each node has 32GB operating memory. The operating system is the Red Hat Enterprise Linux.

We used XQuery [4] compiler to generate sufficiently big execution plan which consists of around 170 operators. As input data we used 10MB XML file. The results do not include the time needed to read the input data from hard drive. Each time in the chart is calculated as the median of 5 measurement of the same test to eliminate inaccuracy. To measure scalability, we were increasing the number of concurrent queries exponentially. The label $N \times M$ in the figure means that we ran N times M queries concurrently.

For comparison, we used very simple scheduler, which does not distinguish between immediate and deferred tasks. Each thread keeps its own list of tasks and processes this list in the FIFO order. If there is no task in its own list, it tries to steal a task from other threads in round-robin order. For threads suspending, one global semaphore is used. The value of the semaphore is kept the same as the number of tasks. The size of packets is constant without respecting the cache size.

The result is shown in the Figure 3. Our new implementation of the scheduler improves the performance about 8 – 12% according to the load of the system. The experiments show that the described ideas are correct and help improve the efficiency of the data processing.

7 Conclusion

In this paper, we proposed the architecture of task scheduler for the systems, which process data streams in parallel environment. We also implemented this scheduler and measured its behavior in the Bobox system. The new scheduler is about 8 – 12 percent more efficient and is NUMA aware. Moreover the techniques,

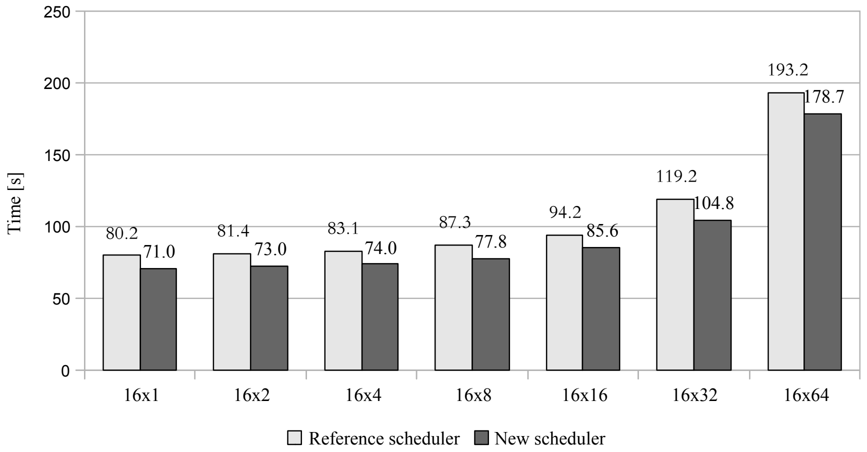


Fig. 3. The effectivity of the new implementation of the scheduler. $N \times M$ means that we ran N times M queries concurrently.

we described, can be used not only in our system, but it is possible to adopt them easily by other similar systems, which contain some kind of task scheduling mechanism.

On the other hand, many challenges still remain. One of them is the analysis of runtime characteristics of the request and dynamic changes of some parameters according to the characteristics to achieve better results. This is related mainly to the size of packets, which is hard to estimate statically without the knowledge of exact behavior of operators.

We not only plan to do optimizations for the size of cache but also for its hierarchy. This means that we want to take cache sharing into account as well. This brings some issues because it is hard to say which threads share caches and which do not since the operating system can change the affinity of the threads automatically. The straightforward solution of manual setting the affinities of threads directly to logical processor does not work well, primarily when processors have the Hyper-Threading technology. The reason is that the thread scheduler of the operating system cannot perform any load balancing among logical processors, which might yield to performance degradation.

References

1. D.J. Abadi, Y. Ahmad, M. Balazinska, U. Cetintemel, M. Cherniack, J.H. Hwang, W. Lindner, A.S. Maskey, A. Rasin, E. Ryvkina, et al. The design of the borealis stream processing engine. In *Second Biennial Conference on Innovative Data Systems Research (CIDR 2005), Asilomar, CA*. Citeseer, 2005.

2. A. Arasu, B. Babcock, S. Babu, M. Datar, K. Ito, I. Nishizawa, J. Rosenstein, and J. Widom. STREAM: the stanford stream data manager (demonstration description). In *Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, page 665. ACM, 2003.
3. B. Babcock, S. Babu, M. Datar, R. Motwani, and D. Thomas. Operator scheduling in data stream systems. *The International Journal on Very Large Data Bases*, 13(4):333–353, 2004.
4. D. Bednárek. *Bulk Evaluation of User-Defined Functions in XQuery.* PhD thesis, Faculty of Mathematics and Physics, Czech Republic, 2009.
5. D. Bednárek and J. Dokulil. Tri Query: Modifying XQuery for RDF and Relational Data. In *2010 Workshops on Database and Expert Systems Applications*, pages 342–346. IEEE, 2010.
6. D. Bednárek, J. Dokulil, J. Yaghob, and F. Zavoral. The Bobox Project - A Parallel Native Repository for Semi-structured Data and the Semantic Web. *TAT 2009 - IX. Informačné technológie - aplikácie a teória*, pages 44–59, 2009.
7. David Bednarek, Jiri Dokulil, Jakub Yaghob, and Filip Zavoral. Using methods of parallel semi-structured data processing for semantic web. *Advances in Semantic Processing, International Conference on*, 0:44–49, 2009.
8. S. Boag, D. Chamberlin, M.F. Fernández, D. Florescu, J. Robie, J. Siméon, and M. Stefanescu. XQuery 1.0: An XML query language. *W3C working draft*, 15, 2002.
9. R. Chandra. *Parallel programming in OpenMP.* Morgan Kaufmann, 2001.
10. J. Cieslewicz, W. Mee, and K.A. Ross. Cache-conscious buffering for database operators with state. In *Proceedings of the Fifth International Workshop on Data Management on New Hardware*, pages 43–51. ACM, 2009.
11. A. Duran, J. Corbalán, and E. Ayguadé. Evaluation of OpenMP task scheduling strategies. In *Proceedings of the 4th international conference on OpenMP in a new era of parallelism*, pages 100–110. Springer-Verlag, 2008.
12. M.A. Hammad, M.F. Mokbel, M.H. Ali, W.G. Aref, A.C. Catlin, A.K. Elmagarmid, M. Eltabakh, M.G. Elfeky, T.M. Ghanem, R. Gwadera, et al. Nile: A query processing engine for data streams. In *Data Engineering, 2004. Proceedings. 20th International Conference on*, page 851. IEEE, 2004.
13. Q. Jiang and S. Chakravarthy. Scheduling strategies for processing continuous queries over streams. *Key Technologies for Data Management*, pages 16–30, 2004.
14. D. Maier et al. NiagaraST. `http://datalab.cs.pdx.edu/niagara/`, 2010. [Online; accessed 21-December-2010].
15. E. Prud'Hommeaux, A. Seaborne, et al. SPARQL query language for RDF. *W3C working draft*, 4, 2006.
16. J. Reinders. *Intel threading building blocks.* O'Reilly, 2007.
17. Ali A. Safaei and Mostafa S. Haghjoo. Parallel processing of continuous queries over data streams. *Distrib. Parallel Databases*, 28:93–118, December 2010.

Exploration of Semantic Spaces Obtained from Czech Corpora

Lubomír Krčmář, Miloslav Konopík, and Karel Ježek

Department of Computer Science and Engineering,
University of West Bohemia, Plzeň, Czech Republic
{lkrcmar, konopik, jezek_ka}@kiv.zcu.cz

Abstract. This paper is focused on semantic relations between Czech words. Knowledge of these relations is crucial in many research fields such as information retrieval, machine translation or document clustering. We obtained these relations from newspaper articles. With the help of LSA[1], HAL[2] and COALS[3] algorithms, many semantic spaces were generated. Experiments were conducted on various settings of parameters and on different ways of corpus preprocessing. The preprocessing included lemmatization and an attempt to use only "open class" words. The computed relations between words were evaluated using the Czech equivalent of the Rubenstein-Goodenough test. The results of our experiments can serve as the clue whether the algorithms (LSA, HAL and COALS) originally developed for English can be also used for Czech texts.

Keywords: Information retrieval, Semantic space, LSA, HAL, COALS, Rubenstein-Goodenough test

1 Introduction

There are many reasons to create a net of relations among words. As with many other research groups, we are trying to find a way how to facilitate information retrieval. Question answering and query expansion are our main interests. We try to employ nets of words in these fields of research. Not only can people judge whether two words have something in common (they are related) or that they are similar (they describe the same idea). Computers with their computational abilities can also make some conclusions about how words are related with each other. Their algorithms exploit the Harris distributional hypothesis [1], which assumes that terms are similar to the extent to which they share similar linguistic contexts. Algorithms such as LSA, HAL and novel COALS were designed to compute the lexical relations automatically. Our belief is that these methods have not yet been sufficiently explored for other languages than English. A

[1] Latent Semantic Analysis

[2] Hyperspace Analogue to Language

[3] the Correlated Occurrence Analogue to Lexical Semantics

great motivation for us was also the S-Space package [2]. The S-Space package is a freely available collection of implemented algorithms dealing with text corpora. LSA, HAL and COALS algorithms are included. Our paper evaluates the applicability of these popular algorithms to Czech corpora.

The rest of the paper is organized as follows. The following section deals with related works. The next section describes the way we created semantic spaces for ČTK[4] corpora. Our experiments and evaluations using the RG benchmark are presented in section 4. In the last section we summarize our experiments and present our future work.

2 Related works

The principles of LSA can be found in [3], the HAL algorithm is described in [4]. A great inspiration for us was a paper about the COALS algorithm [5], where the power of COALS, HAL and LSA is compared. The Rubenstein-Goodenough [6] benchmark and some other similar tests such as Miller-Charles [7] or Word-353 are performed. The famous TOEFL[5] or ESL[6] are also included in the evaluation.

We also come from a paper written by Paliwoda [8] where the Rubenstein-Goodenough (RG) test translated into Polish was used. Alternative ways of evaluating semantic spaces can be found in [9] by Bullnaria and Levy.

Different methods which judge how some words are related exploit lexical databases such as WordNet [10]. There are nouns, verbs, adjectives and adverbs grouped in sets of synonyms called synsets in WordNet. Each synset expresses a distinct concept and all the concepts are interlinked with relations including hypernymy, hyponymy, holonymy or meronymy. Although lexical-based methods are popular and still under review, we have decided to follow the fully automatic methods.

3 Generation of Semantic Spaces

The final form of semantic space is firstly defined by the quality of the corpus used [9] and secondly by the selection of algorithm. The following chapter applies to the features of our corpus and also describes the ways we preprocessed it. The next chapter is focused on parameter settings of LSA, HAL and COALS.

3.1 Corpus and corpus preprocessing

The ČTK 1999 corpus, which consists of newspaper articles, was used for our experiments. The ČTK corpus is one of the largest Czech corpuses we work with in our department. For lemmatization, Hajic's tagger for the Czech language was used [11].

[4] Česká Tisková Kancelář (Czech News Agency)

[5] Test of English as a Foreign Language

[6] English as a Second Language

There was no further preprocessing of input texts performed. Finally, 4 different input files[7] for the S-Space package were used. The first input file contained plain texts of the ČTK corpora. The second one contained plain text without stopwords. Pronouns, prepositions, conjunctions, particles, interjections and punctuation[8] were considered as stopwords in our experiments. That means that removing stopwords from the text in our paper is the same as keeping only open class words in the text. The third file contained lemmatized texts of the ČTK corpora. And the last file contained lemmatized ČTK corpora without stopwords. Statistics on the texts of the corpus are depicted in Table 1, statistics on texts without stopwords are depicted in Table 2 respectively.

Table 1. ČTK corpus statistics

	Plain texts	Lemmatized texts
Documents' count	130,956	
Tokens' count	35,422,517	
Different tokens' count	579,472	291,090
Tokens' count occurring more than once	35,187,747	35,296,478
Different tokens' count occurring more than once	344,702	165,051

Table 2. ČTK corpus statistics, stopwords removed

	Plain texts	Lemmatized texts
Documents' count	130,956	
Tokens' count	22,283,617	
Different tokens' count	577,297	290,036
Tokens' count occurring more than once	22,049,467	22,158,048
Different tokens' count occurring more than once	343,147	164,467

3.2 Settings of algorithms

The LSA principle differs essentially from HAL and COALS. While HAL and COALS are window-based, LSA deals with passages of texts. The passage of text is presented by a whole text of any article of the ČTK corpus in our case.

[7] Each file contained each document of the corpora. One file line corresponds with one distinct document.

[8] Punctuation is rather a token than a word. It was removed while it is not important for the LSA algorithm.

Both LSA and COALS exploit untrivial mathematical operation SVD[9] while HAL does not. COALS simply combines some HAL and LSA principles [5].

The S-Space package provides default settings for its algorithms. The settings are based on previous research. The default settings of parameters are depicted in Table 3. We tried to change some values of parameters because of the Czech language of our texts. The Czech language differs from English especially in the number of forms for one word and in word order, which is as not strictly fixed as in English. Therefore, there are more different terms[10] for Czech language texts. Since the algorithms are sensitive to the term occurrence, this is one of the reasons[11] we tried to remove low occurring words.

Another parameter we observed is HAL's window size. It was expected that the more terms for the Czech language meant the smaller window size would be more appropriate.

The last parameters we changed from defaults were HAL and COALS retained columns' counts. We reduced the dimensionality of spaces in this way by setting the reduction property to the values adopted from [4]. As a consequence, columns with high entropy were retained. To reduce the dimensionality of the COALS algorithm, the impact of SVD was also tested.

Table 3. The default settings of algorithms provided by the S-Space package

Algorithm	property	value
LSA	term-document matrix transform	log-entropy weighting
	the number of dimensions in the semantic space	300
HAL	window size	5
	weighting	linear weighting
	retain property	retain all columns
COALS	retain property	retain 14,000 columns
	window size	4
	reduce using SVD	no

4 Evaluation of Semantic Spaces

Several approaches exist to evaluate semantic spaces as noticed in section 2. Unfortunately, most of the standard benchmarks are suitable only for English. To the best of our knowledge, there is no similar benchmark to the Rubenstein-Goodenough (RG) test or to the Miller-Charles test for the Czech language. Therefore we have decided to translate RG test into Czech.

[9] Singular Value Decomposition

[10] One word in two forms means two terms in this context.

[11] Another reason is to decrease the computation costs.

The following chapter describes the origination of the Czech equivalent of the RG test. The next chapter comprises our results on this test for many generated semantic spaces.

4.1 Rubenstein-Goodenough test

The RG test comprises pairs of nouns with corresponding values from 0 to 4 indicating how much words in pairs are related. The powers of relations were judged by 51 humans in 1965. There were 65 word pairs in the original English RG test.

The translation of the original English RG test into Czech was performed by a Czech native speaker. The article by Pilot [12] describing the original meanings of the RG test's words was exploited. The resulting translation of the test was corrected by 2 Czech native speakers who are involved in information retrieval.

After our translation of the RG test into Czech, 62 pairs are left. We had to remove the "midday-noon", "cock-rooster" and "grin-smile" pairs because we couldn't find any appropriate and different translations for both words of these pairs in Czech. Our Czech RG test[12] was evaluated by 24 Czech native speakers with differing education, age and sex. Pearson's correlation between Czech and English evaluators is 0.94.

A particular word we removed from our test before comparing it with semantic spaces is "crane". The Czech translation of this word has 3 different meanings. Furthermore, only one of these meanings was commonly known by the people who participated in our test. Therefore, another 3 pairs disappeared: "bird-crane", "crane-implement" and "crane-rooster". A similar ambiguate word is the Czech translation of "mound", which was also used in a different meaning in the corpus. We removed it with these 4 pairs: "hill-mound", "cemetery-mound", "mound-shore", "mound-stove". In the end, 55 word pairs were left in our test.

Another issue we had to face was the low occurrence of the RG test's words in our corpus. Therefore, we tried to remove the least frequent words of the RG test in sequence and the pairs which they appear in as a consequence. In the end it turned out that especially this step showed us that the relations obtained from S-Space algorithms correlate with human judgments quite well. To evaluate which of the semantic spaces best fits with human judgments the standard Pearson's correlation coefficient was used.

4.2 Experiments and results

We created many semantic spaces with the LSA, HAL and COALS algorithms. Cosine similarity was used to evaluate whether two words are related in semantic spaces. Other similarity metrics did not work well.

The obtained results for the different semantic spaces are depicted in Table 4 for plain texts of the ČTK corpora and in Table 5 for lemmatized texts. The

[12] Available at `http://home.zcu.cz/~lkrcmar/RG/RG-ENxCZ.pdf`

best 2 scores in Table 4 and the best 3 scores in Table 5 are highlighted for each tested set of pairs in our RG test.

Table 4. Correlation between values for pairs obtained from different semantic spaces and the Czech Rubenstein-Goodenough test. The word pairs containing low occurring words in the corpora were omitted in sequence (o-27 means 27 pairs out of the original 65 were omitted while computing the correlation). N - no stopwords, m2 - words occurring more than once in the corpora are retained for the computation, s1 - window size = 1, d2 - reduce to 200 dimensions using the SVD.

Semantic space	o-27	o-29	o-32	o-35	o-37	o-44	o-51
LSA_m2	0,26	0,25	0,27	0,33	0,35	0,36	0,24
N_LSA	0,28	0,28	0,29	0,33	0,33	0,33	0,16
N_LSA_m2	0,27	0,26	0,29	0,33	0,30	0,32	0,11
HAL_m2	0,20	0,19	0,24	0,28	0,25	0,24	0,14
HAL_m2_s1	0,12	0,11	0,18	0,19	0,14	0,06	0,04
HAL_m2_s2	0,17	0,18	0,25	0,30	0,25	0,18	0,15
N_HAL_m2	0,36	0,38	0,39	0,43	0,43	0,44	0,44
N_HAL_m2_s1	0,39	0,41	0,43	0,47	0,46	0,48	0,53
N_HAL_m2_s2	0,40	0,42	0,44	0,48	0,48	0,49	0,53
COALS_m2	**0,43**	**0,45**	**0,48**	**0,52**	**0,54**	**0,57**	**0,62**
COALS_m2_d2	0,28	0,30	0,30	0,35	0,38	0,39	0,42
COALS_m2_d4	0,17	0,18	0,18	0,19	0,21	0,27	0,32
N_COALS_m2	0,42	0,43	**0,46**	**0,50**	**0,53**	0,54	**0,59**
N_COALS_m2_d2	0,31	0,27	0,25	0,35	0,31	0,23	0,34
N_COALS_m2_d4	**0,43**	**0,44**	0,45	0,50	0,51	0,51	0,57

It turned out we do not have to take into account words which occur only once in our corpora. It saves computing time without a negative impact on the results. This is the reason most of our semantic spaces are computed omitting words which occur only once.

The effect of omitting stopwords is very small for the LSA and the COALS algorithms. However, the HAL algorithm scores are affected a lot (compare HAL and N_HAL in Tables 4 and 5). This difference in results can be caused by the fact that LSA does not use any window and works with whole texts. The COALS algorithm may profit from using the correlation principle[5] that helps it to deal with stopwords.

The scores in our tables show that especially the COALS method is very successful. The best scores are achieved by COALS for the plain texts, and COALS scores for the lemmatized texts are also among the best ones (compare Table 4 and 5).

The HAL method is also very successful. Furthermore, the best score of 0.72 is obtained using the HAL method on lemmatized data without stopwords (see Table 5). It turns out that HAL even outperforms COALS when only pairs

Table 5. Correlation between values for pairs obtained from different semantic spaces and the Czech Rubenstein-Goodenough test. The word pairs containing low occurring words in the corpora were omitted in sequence (o-27 means 27 pairs out of the original 65 were omitted while computing the correlation). N - no stopwords, m2 - words occurring more than once in the corpora are retained for the computation, s1 - window size = 1, r14 - only 14,000 columns retained, d2 - reduce to 200 dimensions using the SVD.

Semantic space	o-14	o-19	o-24	o-27	o-29	o-32	o-35	o-37	o-44	o-51
LSA	0,19	0,22	0,25	0,33	0,35	0,35	0,44	0,47	0,48	0,47
LSA_m2	0,15	0,19	0,22	0,30	0,33	0,33	0,41	0,46	0,47	0,41
N_LSA	0,16	0,18	0,20	0,30	0,33	0,36	0,44	0,46	0,47	0,37
N_LSA_m2	0,17	0,19	0,21	0,32	0,36	0,37	0,43	0,47	0,47	0,39
HAL	0,35	0,44	0,45	0,47	0,48	0,53	0,57	0,54	0,57	0,41
HAL_m2	0,35	0,44	0,45	0,47	0,48	0,53	0,57	0,53	0,57	0,41
HAL_m2_s1	0,37	0,41	0,41	0,41	0,42	0,48	0,50	0,47	0,49	0,34
HAL_m2_s2	0,45	0,51	0,52	0,54	0,57	**0,62**	**0,68**	**0,64**	**0,67**	0,56
HAL_m2_s10	0,26	0,41	0,43	0,48	0,48	0,54	0,56	0,52	0,56	0,35
HAL_m4	0,35	0,44	0,45	0,47	0,48	0,53	0,57	0,53	0,57	0,41
HAL_r14	0,40	0,47	0,48	0,50	0,52	**0,58**	**0,61**	0,57	**0,62**	0,48
HAL_r7	0,39	0,46	0,46	0,48	0,50	0,55	0,58	0,54	0,58	0,43
N_HAL_m2	0,22	0,26	0,29	0,34	0,35	0,33	0,36	0,37	0,39	0,26
N_HAL_m2_s1	0,43	0,45	0,49	0,52	0,55	0,55	**0,62**	**0,64**	**0,68**	**0,72**
N_HAL_m2_s2	0,34	0,37	0,40	0,44	0,48	0,48	0,54	0,55	0,61	**0,61**
COALS	**0,52**	0,53	**0,55**	**0,54**	**0,57**	0,54	0,58	0,55	0,57	0,61
COALS_m2	**0,52**	**0,53**	**0,55**	**0,55**	**0,57**	0,54	0,58	0,55	0,57	**0,61**
COALS_m2_r7	0,52	**0,53**	0,53	0,52	0,54	0,53	0,56	0,55	0,56	0,59
COALS_m2_d2	0,22	0,22	0,42	0,40	0,38	0,40	0,43	0,40	0,48	0,42
COALS_m2_d4	0,32	0,35	0,40	0,41	0,43	0,46	0,41	0,42	0,40	0,56
COALS_m4	0,48	0,48	0,50	0,50	0,52	0,50	0,54	0,52	0,53	0,55
N_COALS_m2	**0,53**	**0,54**	**0,57**	**0,56**	**0,59**	**0,56**	0,60	**0,59**	0,59	0,60
N_COALS_m2_d2	0,26	0,27	0,22	0,28	0,31	0,32	0,41	0,45	0,45	0,55
N_COALS_m2_d4	0,32	0,34	0,38	0,43	0,46	0,45	0,51	0,51	0,56	0,53

containing only very common words are left. On the other hand, this shows the strength of COALS when also considering low occurring words in our corpora.

It turned out that the LSA algorithm is not as effective as the other algorithms in our experiments. Our hypothesis is that scores of LSA would be better when experimenting with larger corpora such as Rohde [5]. However, the LSA scores also improve when considering only common words. This Figure 1 shows the performance of the 3 tested algorithms for the best settings found.

Our results differ from scores of tests evaluated on English corpora and performed by Rohde [5]. His scores for HAL are much lower than ours. On the other hand, his scores for LSA are higher. Therefore, we believe that the performances of the algorithms are language dependent.

The last Figure 2 in our paper compares human and HAL judgments about the relatedness of 14 pairs containing the most common words from the RG word list in the ČTK corpora. The English equivalents to the Czech word pairs are listed in Table 6. We can notice the pairs which spoil the scores of the tested algorithms in the graph. The graph also shows the difference in human and machine judgments. The pair "automobile-car" is less related than "food-fruit" for the algorithms than for humans. On the other hand, the words of the pair "coast-shore" are more related for our algorithms than for humans.

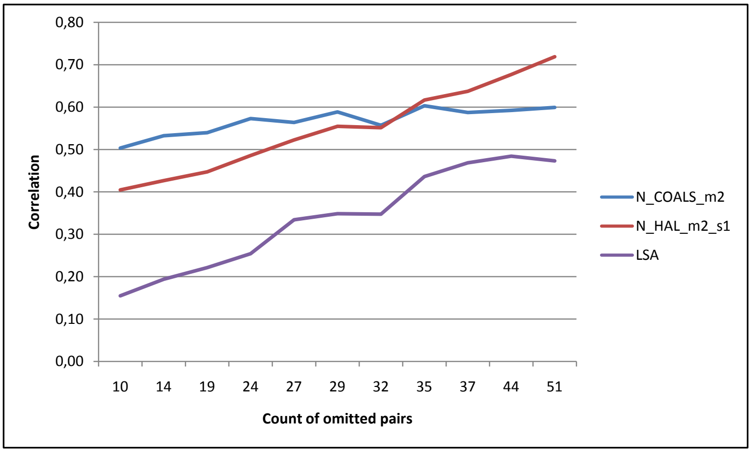


Fig. 1. Graph depicting the performances of LSA, HAL and COALS depending on leaving out rare words in the corpora. Our best settings found for algorithms are chosen.

5 Conclusion

Our experiments showed that HAL and COALS algorithms performed well and better than LSA on the Czech corpora. Our hypothesis based on our results is

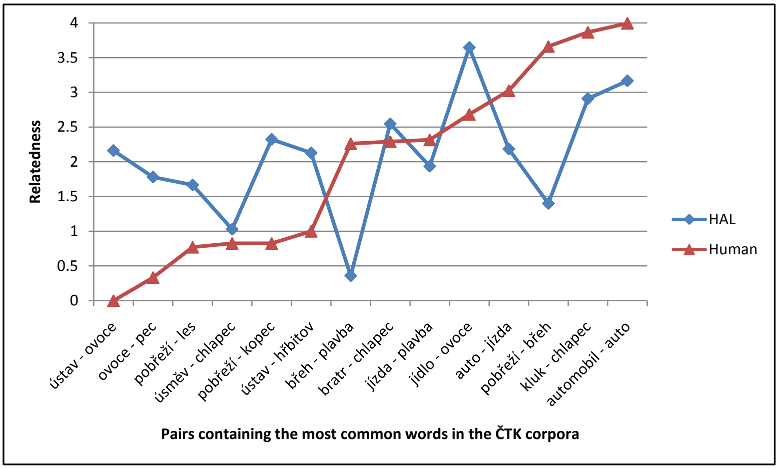


Fig. 2. Graph depicting the comparison between human and HAL judgments (value of Cosine similarity of vectors multiplied by 4 is used) about the relatedness of words in pairs. The pairs from the RG test containing only the most common words in the ČTK corpora are left. Our best HAL setting is chosen. The pairs on the X axis are sorted according to the human similarity score.

Table 6. The English translation of the Czech word pairs in Figure 2

Czech word pair	English equivalent	Czech word pair	English equivalent
ústav - ovoce	asylum - fruit	bratr - chlapec	brother - lad
ovoce - pec	fruit - furnace	jízda - plavba	journey - voyage
pobřeží - les	coast - forest	jídlo - ovoce	food - fruit
úsměv - chlapec	grin - lad	auto - jízda	car - journey
pobřeží - kopec	coast - hill	pobřeží - beh	coast - shore
ústav - hřbitov	asylum - cemetery	kluk - chlapec	boy - lad
břeh - plavba	shore - voyage	automobil - auto	automobile - car

that COALS semantic spaces are more accurate for low occurring words, while semantic spaces generated by HAL are more accurate for pairs of words with higher occurrence. Our experiments show that the lemmatization of corpora is the appropriate approach to improve the scores of algorithms. Furthermore, the best scores of correlation were achieved when only the ”open class” words were used.

It turned out that the translation of the original English RG test was not so appropriate for our Czech corpora while it contains words which are not so common in the corpora. However, we believe that when the pairs containing low occurring words were removed, the applicability of the test was improved. The evidence for this is a discovered dependency between the scores of tested algorithms on omitting pairs with low occurring words in them.

We believe that semantic spaces are applicable for the query expansion task which we will focus on in our future work. Apart from this, we are attempting to get some larger Czech corpora for our experiments. We also plan to continue testing the HAL and COALS algorithms, which performed well during our experiments.

Acknowledgment

The work reported in this paper was supported by the Advanced Computer and Information Systems project no. SGS-2010-028. The access to the MetaCentrum supercomputing facilities provided under the research intent MSM6383917201 is also highly appreciated. Finally, we would like to thank the Czech News Agency for providing text corpora.

References

1. Harris, Z. (1954). Distributional structure. (J. Katz, Ed.) *Word Journal Of The International Linguistic Association, 10*(23), 146-162. Oxford University Press.
2. Jurgens and Stevens, (2010). The S-Space Package: An Open Source Package for Word Space Models. In *System Papers of the Association of Computational Linguistics.*
3. Landauer, T., Foltz, P., & Laham, D. (1998). An introduction to latent semantic analysis. *Discourse Processes, 25*(2), 259-284. Routledge.
4. Lund, K., & Burgess, C. (1996). Producing high-dimensional semantic spaces from lexical co-occurrence. *Behav Res Methods Instrum Comput, 28*(2), 203-208 203208.
5. Rohde, D. T., Gonnerman, L., & Plaut, D. (2004). An improved method for deriving word meaning from lexical co-occurrence. *Cognitive Science.*
6. Rubenstein, H., & Goodenough, J. (1965). Contextual correlates of synonymy. *Communications of the ACM, 8*(10), 627-633. ACM Press.
7. Miller, G., & Charles, W. (1991). Contextual Correlates of Semantic Similarity. *Language & Cognitive Processes, 6*(1), 1-28. Psychology Press.
8. Paliwoda-Pękosz, G., Lula, P.: Measures of Semantic Relatedness Based on Wordnet. In: International workshop for PhD students, 2009 Brno. ISBN 978-80-214-3980-1

9. Bullinaria, J., & Levy, J. (2007). Extracting semantic representations from word co-occurrence statistics: a computational study. *Behavior Research Methods, 39*(3), 510-526. Psychonomic Society Publications.
10. George A. Miller. 1995. Miller, G. (1995). WordNet: a lexical database for English. *Communications of the ACM, 38*(11), 39-41. ACM.
11. J. Hajič, A. Böhmová, E. Hajičová, B. Vidová Hladká, The Prague Dependency Treebank: A Three-Level Annotation Scenario. In A. Abeillé (ed.): *Treebanks Building and Using Parsed Corpora.* pp. 103-127. Amsterdam, The Netherlands: Kluwer, 2000.
12. OShea, J., Bandar, Z., Crockett, K., & McLean, D. (2008). Pilot Short Text Semantic Similarity Benchmark Data Set: Full Listing and Description. *Computing.*

Using SVM and Clustering Algorithms in IDS Systems

Peter Scherer, Martin Vicher, Pavla Dráždilová, Jan Martinovič,
Jiří Dvorský,, and Václav Snášel

Department of Computer Science, FEI, VSB – Technical University of Ostrava,
17. listopadu 15, 708 33, Ostrava-Poruba, Czech Republic
{peter.scherer, martin.vicher, pavla.drazdilova, jan.martinovic,
jiri.dvorsky, vaclav.snasel}@vsb.cz

Abstract. Intrusion Detection System (IDS) is a system, that monitors network traffic and tries to detect suspicious activity. In this paper we discuss the possibilities of application of clustering algorithms and Support Vector Machines (SVM) for use in the IDS. There we used K-means, FarthestFirst and COBWEB algorithms as clustering algorithms and SVM as classification SVM of type 1, known too as C-SVM. By appropriate choosing of kernel and SVM parameters we achieved improvements in detection of intrusion to system. Finally, we experimentally verified the efficiency of applied algorithms in IDS.

Key words: Intrusion Detection System, K-means, Farthest First Traversal, COBWEB/CLASSIT, SVM, clustering

1 Introduction

Three criteria are important for computer systems security: confidentiality, integrity and availability. Computer security is defined as a protection against threads for these criteria. The major manners of computer security are techniques like user authentication, data encryption, avoiding programming errors and firewalls. They are known as first line of defense. The last line of defense is used *Intrusion Detection System* (IDS). An Intrusion Detection System is software application (device respectively) that monitors network and system activities for malicious attempts, threads or policy violations and produces reports and statistics. Several machine-learning paradigms including soft computing approach [2], neural networks and fuzzy inference system [11], genetic algorithms [14], Bayesian network, matrix factorization approach [16], multivariate adaptive regression splines etc. have been investigated for the design of IDS. In this paper we investigate and evaluate the performance of Farthest First Traversal, K-means, COBWEB/CLASSIT clustering algorithms and classification via Support Vector Machines. The motivation for using the clustering algorithms and SVM is to improve the accuracy of the Intrusion Detection System.

2 Clustering Algorithms and Their Classification

Cluster analysis is the process of grouping the objects (usually represented as a vector of measurements, or a point in a multidimensional space) so that the objects of one cluster are similar to each other whereas objects of different clusters are dissimilar.

Clustering is the unsupervised classification of objects (observations, data items, instances, cases, patterns, or feature vectors) into groups, *clusters*. In [4] author cite that from a machine learning perspective, clusters correspond to hidden patterns, the search for clusters is unsupervised learning, and the resulting system represents a data concept. Therefore, clustering is unsupervised learning of a hidden data concept.

The applications of clustering often deal with large datasets and data with many attributes. Clustering is related to many other fields. The classic introduction to clustering in pattern recognition is given in [7]. Machine learning clustering algorithms were applied to image segmentation and computer vision [12].

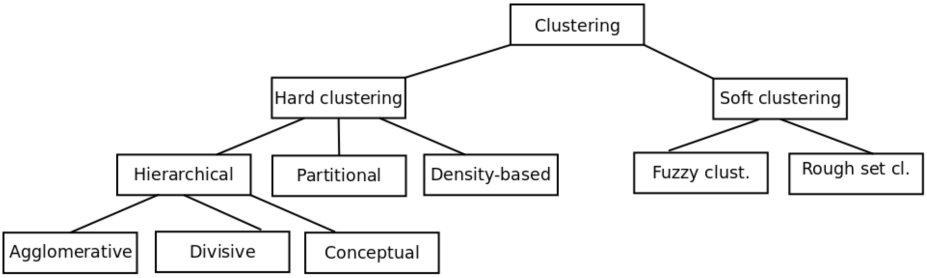


Fig. 1. A taxonomy of clustering approaches.

2.1 Classification of Clustering Algorithms

The various clustering algorithms can be classified according to how they create clusters of objects. Such division of clustering algorithms is shown in Fig. 1.

For our intention of using the clustering algorithms in an IDS, we need algorithms that can determine the jurisdiction of the object X to cluster, even if the object X was not included in the set of objects, from which we generate clusters. For this purpose we chose the algorithms K-means, Farthest First Traversal (they are partitional algorithms) and Cobweb/CLASSIT (this is a conceptual clustering algorithm).

Partitional Algorithms Partitional algorithms divide the objects into several disjoint sets and creates a one level of non-overlapping clusters. But the problem is to determine how many clusters has algorithm detect.

Algorithms of Conceptual Clustering Algorithms of conceptual clustering create by incremental way, the structure of the data by division of observed objects into subclasses. The result of these algorithms is a classification tree. Each node of the tree contains the objects of its child nodes, so root of this tree contains a all objects. According to the above classification are a these algorithms hierarchical, incremental algorithms that combine both – aggregation and division approach.

2.2 Farthest First Traversal

Farthest first traversal (FFT) algorithm is partitional clustering algorithm. This algorithm first select K objects as the centers of clusters and then assign other objects into the cluster (according to measure of dissimilarity to centers of the clusters). The first center of cluster is chosen randomly, the second center of cluster as most dissimilar to first center of cluster and every other center of cluster is chosen as the one whose value of measure of dissimilarity [9] to the previously selected centers of the clusters is greatest.

2.3 K-means

Algorithm K-means, according to the classification above is partitional clustering algorithm. The main idea of the algorithm is to find K centers (one for each cluster) of clusters. The question is, how choose these centers of clusters, because this choice will significantly affect the resulting clusters. The best would be to pick center of cluster least similar to each other. The next step is assign each object from data set to the center of cluster, to which is most similar. Once this occurs, the next step in the classification is to determine the new center of each cluster (centers are derived from clusters of objects). Again, is performed the classification of objects into different clusters according to their dissimilarity [9] with new centers of clusters. These steps are repeated until we find out that centers of clusters no longer change or until is achieved maximum number of repetitions.

2.4 COBWEB/CLASSIT

This incremental clustering algorithm creates a hierarchical structure of clusters by using four operators (operator for creating a new cluster, inserting an object into an existing cluster, union of two clusters into one cluster and splitting cluster into two clusters) [8] and the categorization utility [15]. When processing object into the cluster is always used one of the operators, but always are tested all four operators and categorization utility evaluate distribution of clusters after applying one of the operator. Finally, as the resulting distribution is chosen distribution that was evaluated (by using a categorization utility) as the best.

3 Classification SVM of type 1 (C-SVM) and their parameters

3.1 Support Vector Machines Classifier

Support Vector Machine (SVM) is a preferably technique for linear binary data classification. In [10] authors state that a classification task usually involves separating data into training and testing sets. Each instance in the training set contains one target value (i.e. the class labels) and several attributes (i.e. the features or observed variables). The goal of SVM is to produce a model (based on the training data) which predicts the target values of the test data given only the test data attributes.

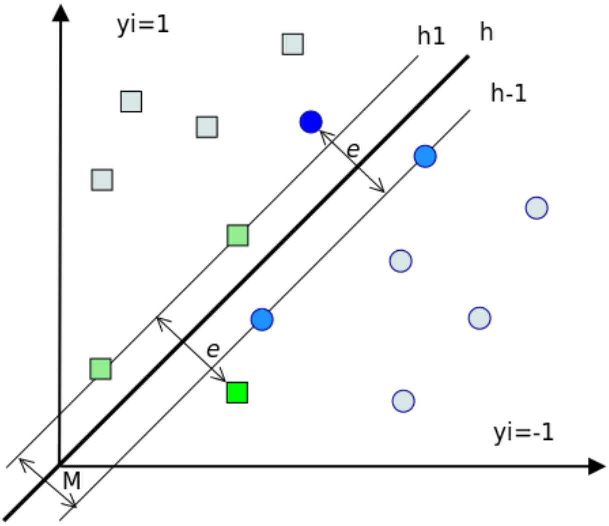


Fig. 2. General linear binary classification case.

Given a binary training set $(\boldsymbol{x}_i, y_i)$, $\boldsymbol{x}_i \in \mathbb{R}^n$, $y_i \in \{-1, 1\}$, $i = 1, \ldots, m$, the basic variant of the SVM algorithm attempts to generate a separating hyper-plane in the original space of n coordinates (x_i parameters in vector $\boldsymbol{x}$) between two distinct classes, Fig. 2. During the training phase the algorithm seeks for a hyper-plane which best separates the samples of binary classes (classes 1 and -1). Let $h_1 : \boldsymbol{wx} + b = 1$ and $h_{-1} : \boldsymbol{wx} + b = 1$ ($\boldsymbol{w}, \boldsymbol{x} \in \mathbb{R}^n, b \in \mathbb{R}$) be possible hyper-planes such that majority of class 1 instances lie above h_1 and majority of class -1 fall below h_{-1}, whereas the elements coinciding with h_1, h_{-1} are hold for Support Vectors. Finding another hyper-plane $h : \boldsymbol{wx} + b = 0$ as the best separating (lying in the middle of h_1, h_{-1}), assumes calculating $\boldsymbol{w}$ and b, i.e. solving the nonlinear convex programming problem. The notion of the best separation can be formulated as finding the maximum margin M that separates the data from both classes. Since $M = 2\left\|\boldsymbol{w}\right\|^{-1}$, maximizing the margin cuts down to minimizing $\|\boldsymbol{w}\|$ Eq.(1).

$$\min_{\boldsymbol{w},b} \frac{1}{2}\left\|\boldsymbol{w}\right\|^2 + C\sum_i \varepsilon_i \tag{1}$$

with respect to: $1 - \varepsilon_i - y_i(\boldsymbol{w}\cdot\boldsymbol{x}_i + b) \leq 0$, $-\varepsilon_i \leq 0$, $i = 1, 2\ldots, m$

Regardless of having some elements misclassified (Fig. 2) it is possible to balance between the incorrectly classified instances and the width of the separating margin. In this context, the positive slack variables ε_i and the penalty parameter C are introduced. Slacks represents the distances of misclassified points to the initial hyper-plane, while parameter C models the penalty for misclassified training points, that trades-off the margin size for the number of erroneous classifications (bigger the C smaller the number of misclassifications and smaller the margin). The goal is to find a hyper-plane that minimizes misclassification errors while maximizing the margin between classes. This optimization problem is usually solved in its dual form (dual space of Lagrange multipliers):

$$\boldsymbol{w}^* = \sum_{i=1}^{m} \alpha_i y_i \boldsymbol{x}_i \tag{2}$$

where $C \geq \alpha_i \geq 0$, $i = 1, \ldots, m$, and where $\boldsymbol{w}^*$ is a linear combination of training examples for an optimal hyper-plane. However, it can be shown that $\boldsymbol{w}^*$ represents a linear combination of Support Vectors $\boldsymbol{x}_i$ for which the corresponding α_i Langrangian multipliers are non-zero values. Support Vectors for which $C > \alpha_i > 0$ condition holds, belong either to h_1 or h_{-1}. Let x_a and x_b be two such Support Vectors ($C > \alpha_a, \alpha_b > 0$) for which $y_a = 1$ and $y_b = -1$. Now b could be calculated from $b^* = 0.5\boldsymbol{w}^*(\boldsymbol{x}_a + \boldsymbol{x}_b)$, so that classification (decision) function finally becomes:

$$f(\boldsymbol{x}) = \operatorname{sgn}\sum_{i=1}^{m} \alpha_i y_i (\boldsymbol{x}_i \cdot \boldsymbol{x}) + b^* \tag{3}$$

To solve non-linear classification , one can propose the mapping of instances to a so-called feature space of very high dimension: $\varphi : \mathbb{R}^n \to \mathbb{R}^d$, $n \ll d$ i.e. $\boldsymbol{x} \to \varphi(\boldsymbol{x})$. The basic idea of this mapping into a high dimensional space is to transform the non-linear case into linear and then use the general algorithm already explained above Eqs. (1), (2), and (3). In such space, dot-product from Eq. (3) transforms into $\varphi(\boldsymbol{x}_i) \cdot \varphi(\boldsymbol{x})$. A certain class of functions called *kernels* [6] for which $k(\boldsymbol{x}, \boldsymbol{y}) = \varphi(\boldsymbol{x}) \cdot \varphi(\boldsymbol{y})$ holds, are called kernels. They represent dot-products in some high dimensional dot-product spaces (feature spaces), and yet could be easily recomputed into the original space. As example was chosen a Radial Basis Function Eq. (4), also known as Gaussian kernel [1], and was one of implemented kernels in the experimenting procedure.

$$k(\boldsymbol{x}, \boldsymbol{y}) = \exp(-\gamma\left\|\boldsymbol{x} - \boldsymbol{y}\right\|^2) \tag{4}$$

Now Eq. (3) becomes:

$$f(\boldsymbol{x}) = \operatorname{sgn} \sum_{i=1}^{m} \alpha_i y_i k(\boldsymbol{x}_i \cdot \boldsymbol{x}) + b^* \tag{5}$$

After removing all training data that are not Support Vectors and retraining the classifier, the same result would be obtained [6] by applying the function above. Thus, one depicted, Support Vectors could replace the entire training set, which is the central idea of SVM implementation.

4 Experiments

The data used for training and testing was prepared by the Agency DARPA intrusion detection evaluation program in 1998 at MIT Lincoln Labs [13]. Experiments were performed on a collection containing five pairs of data sets: the learning set (5092 vectors of 42 attributes) and testing set (6890 vectors of 42 attributes). Each pair represents a learning and testing data for one type of five classes of network attacks. Individual vectors describing the network traffic are described by 41 attributes (range $0 - 1$, is therefore not necessary to normalization). The 42$^{\text{nd}}$ attribute was used in learning process. The attribute determines type of network attack in the question. In the case of testing, the existence of the attribute was neglected. We measure only classification accuracy of the vector, that describes the network attack.

4.1 Classification Using SVM type 1 (C-SVM)

It is necessary to determine the appropriate combination of parameters C and γ for better efficiency. In our experiment, the parameter C is in the range of 2^{-5} and 2^{15} in increments of powers of 2 and a parameter γ is in the range of 2^{-15} and 2^{3} in increments of powers of 2. We used 110 combinations of parameters C γ in total. In the case of same results of prediction with different parameters C and γ, the combination of parameters with the lowest time-intensive calculation model was chosen. In Tables 1,2, 3, and 4 is possible to see the best result combination.

The four most utilized kernel functions (linear, polynomial, RBF and sigmoid) was used for process of learning. As technology, we used library LibSVM [5].

4.2 Classification with Algorithm Farthest First Traversal

During experiments with the algorithm Farthest First Traversal we tried to reveal the effect of number of generated clusters on success rate of the classification of network traffic, and on training time. The measure used by this algorithm was cosine measure. Tables 5 and 6 shows results of each experiments with algorithm FFT. Of these it is possible to deduce that the time of training increases with the number of generated clusters. We tried to optimize this algorithm by using data structure KD-tree. Training time of this algorithm with and without using

Table 1. Classification using linear kernel.

Attack type	Training time (s)	C	γ	Accuracy (%)
Normal	0.71	2^{-1}	2^{-1}	99.55
Probe	0.25	2^{3}	2^{-1}	99.81
DOS	0.35	2^{7}	2^{-3}	99.81
U2R	0.17	2^{3}	2^{-3}	99.80
R2L	0.35	2^{5}	2^{-5}	99.64

Table 2. Classification using polynomial kernel.

Attack type	Training time (s)	C	γ	Accuracy (%)
Normal	0.78	2^{13}	2^{-7}	99.83
Probe	0.24	2^{-3}	2^{-1}	99.81
DOS	0.47	2^{9}	2^{-5}	97.18
U2R	0.16	2^{15}	2^{-5}	99.80
R2L	0.24	2^{15}	2^{-5}	99.71

Table 3. Classification using RBF kernel.

Attack type	Training time (s)	C	γ	Accuracy (%)
Normal	0.88	2^{1}	2^{-3}	99.87
Probe	0.26	2^{5}	2^{-5}	99.90
DOS	0.29	2^{15}	2^{-7}	99.88
U2R	0.18	2^{9}	2^{-3}	99.83
R2L	0.37	2^{13}	2^{-7}	99.75

Table 4. Classification using sigmoid kernel.

Attack type	Training time (s)	C	γ	Accuracy (%)
Normal	0.95	2^{5}	2^{-5}	99.58
Probe	0.38	2^{7}	2^{-5}	99.88
DOS	0.43	2^{15}	2^{-9}	99.83
U2R	0.20	2^{5}	2^{-3}	99.83
R2L	0.42	2^{11}	2^{-7}	99.65

of KD-tree is shown in Tables 5 and 6. As you can see in the Tables 5 and 6 training time of this algorithm with using KD-tree was reduced by almost half. Table 7 presents the results of the algorithm FFT with using a KD-tree for each class of attack.

Table 5. Results of algorithm FFT for class of attack Normal without using KD-Tree.

Number of clusters	Training time (s)	Accuracy (%)
10	2.99	74.82
20	6.89	74.73
30	8.42	81.86
40	12.72	77.90
50	15.21	77.29
100	25.24	82.03

Table 6. Results of algorithm FFT for class of attack Normal with using KD-Tree.

Number of clusters	Training time (s)	Accuracy (%)
10	1.64	74.82
20	5.38	74.73
30	4.54	81.86
40	5.96	77.90
50	7.51	77.29
100	16.31	82.03

4.3 Classification with Algorithm K-means

During experiments with algorithm K-means we tried to reveal the influence of the number of generated clusters on training time and success rate of the network traffic classification. The measure that was used by this algorithm was cosine measure. In Tables 8, 9 and 10 are shown results for each experiment. Of these it is possible to deduce that the time of training is increasing with the number of generated clusters. We tried to optimize this algorithm by using data structure KD-tree. Training time of this algorithm with and without using of KD-tree is shown in Tables 8 and 9. As you can see in the Tables 8 and 9, training time of this algorithm with using KD-tree not declined as significantly as at algorithm FFT. For certain number of generated clusters was training time even worse than at algorithm without using KD-tree. This is due overhead of

Table 7. Results of algorithm FFT for each class of attack with using KD-Tree.

Attack type	Training time (s)	Accuracy (%)
Normal	5.96	84.92
Probe	5.94	98.77
DOS	6.18	82.64
U2R	5.85	95.04
R2L	5.99	99.27

creating KD-tree in each iteration of the algorithm and for a small number of generated clusters is more effective search cluster, where object fall, sequentially than by using KD-tree. Table 10 presents the results of algorithm K-means using a KD-tree for each class of attack.

Table 8. Results of algorithm K-means for class of attack Normal without using KD-Tree.

Number of clusters	Training time (s)	Accuracy (%)
10	29.53	94.71
20	46.69	99.93
30	60.89	98.64
40	74.88	99.62
50	82.24	99.46
100	147.70	98.27

4.4 Classification with Algorithm COBWEB/CLASSIT

To achieve the best success rate is necessary to determine values of parameters Acuity and Cutoff. These parameters must be selected manually and is not known method how select the best combination. Based on experiments with the values of these parameters, when the values for the parameter Acuity were changed in the interval 0.225 to 0.01 with step 0.025 with the constant value of parameter Cutoff 0.1 and experiments when parameter Acuity had constant value 0.1 and values of parameter Cutoff were changed in the interval $0.1 - 1$ with step 0.1. We have chosen values for parameter Acuity 0.1 and for parameter Cutoff 0.6. Table 11 shown the results of the algorithm COBWEB/CLASSIT for each class of attack.

Table 9. Results of algorithm K-means for class of attack Normal with using KD-Tree.

Number of clusters	Training time (s)	Accuracy (%)
10	36.21	94.71
20	49.83	99.93
30	56.92	98.64
40	67.88	99.62
50	71.20	99.46
100	107.68	98.27

Table 10. Results of algorithm K-means for each class of attack with using KD-Tree.

Attack type	Training time (s)	Accuracy (%)
Normal	71.80	99.46
Probe	79.14	98.19
DOS	98.59	99.91
U2R	95.04	99.97
R2L	101.11	97.46

Table 11. Results of algorithm COBWEB/CLASSIT for each class of attack.

Attack type	Training time (s)	Accuracy (%)
Normal	284.72	83.73
Probe	356.98	97.79
DOS	260.07	83.12
U2R	265.33	93.58
R2l	216.78	97.92

Table 12. Classification using SVM.

Attack type	SVM kernel			
	linear	polynomial	RBF	sigmoid
Normal	99.550	99.830	99.870	99.580
Probe	99.810	99.810	99.900	99.880
DOS	99.810	97.180	99.880	99.830
U2R	99.800	99.800	99.830	99.830
R2L	99.640	99.710	99.750	99.650
Average	99.722	99.266	99.846	99.754

Table 13. Classification using clustering algorithm.

Attack type	FFT	K-means	COBWEB/CLASSIT
Normal	84.92	99.46	83.73
Probe	98.77	98.19	97.79
DOS	82.64	99.91	83.12
U2R	95.04	99.97	93.58
R2L	99.27	97.46	97.92
Average	92.128	98.998	91.228

5 Conclusion

In this paper we have described the method for the illustrated prediction accuracy by using clustering algorithms and SVM in the IDS. In Table 13 for each used algorithm is shown success rate for each class of attack. The best average success rate has SVM algorithm, more than 99% (best of all is algorithm SVM that is using the RBF kernel, it has a success rate 99.722%). The average success rate of other algorithms was between 91.228% and 98.998%. It will be useful to compare these two methods on other document collections. In our future work we will investigate other kernel functions to search for better attacks prediction in the IDS, SVM paralelization and optimalization clustering algorithms.

Acknowledgment

This work is partially supported by Grant of Grant Agency of Czech Republic No. 205/09/1079, and SGS, VSB – Technical University of Ostrava, Czech Republic, under the grant No. SP2011/172.

References

1. S. Abe. Support Vector Machines for pattern classification. London, Springer, 2005.
2. A. Abraham and R. Jain. Soft Computing Models for Network Intrusion Detection Systems. Classification and Clustering for Knowledge Discovery Studies in Computational Intelligence, p. 191–207, 2005.
3. B. Al-Shboul and S.-H. Myaeng. Initializing k-means using genetic algorithms, 2009.
4. P. Berkhin. A Survey of Clustering Data Mining Techniques. Grouping Multidimensional Data, p. 25–71, 2002.
5. Chih-Chung Chang and Chih-Jen Lin. LIBSVM: a library for support vector machines, 2001 `http://www.csie.ntu.edu.tw/~cjlin/libsvm`
6. N. Cristiani and J. Shawe-Taylor. An Introduction to Support Vector Machines and other kernel-based learning methods. Cambridge, Cambridge University Press, 2000.
7. R. Duda and P. Hart. Pattern Classification and Scene Analysis. John Wiley & Sons, New York, 1973.

8. D. H. Fisher. Knowledge Acquisition Via Incremental Conceptual Clustering. Kluwer Academic Publisher, 1987.
9. G. Gan, C. Ma, and J. Wu. Data Clustering Theory. Algorithms and Applications. ASASIAM, 2007.
10. C. Hsu, C. Chang and C. Lin. A Practical Guide to Support Vector Classification, journal Bioinformatics, 2003.
11. S. Chavan, K. Shah, N. Dave, S. Mukherjee, A. Abraham and S. Sanyal. Adaptive Neuro-Fuzzy Intrusion Detection Systems, International Conference on Information Technology: Coding and Computing (ITCC'04), 2004.
12. A. K. Jain and P.J. Flynn. Image segmentation using clustering. In Advances in Image Understanding: A Festschrift for Azriel Rosenfeld, IEEE Press, 65–83, 1996.
13. MIT Lincoln Laboratory `http://www.ll.mit.edu/IST/ideval/`
14. S. Owais, V. Snasel, P. Kromer and A. Abraham. Survey: Using Genetic Algorithm Approach in Intrusion Detection Systems Techniques, p. 300–307, Computer Information Systems and Industrial Management Applications, 2008.
15. N. Sahoo. Incremental hierarchical clustering of text documents. adviser: Jamie Callan, 2006.
16. V. Snasel, J. Platos, P. Kromer, A. Abraham. Matrix Factorization Approach for Feature Deduction and Design of Intrusion Detection Systems, p. 172–179, The Fourth International Conference on Information Assurance and Security, 2008.

Combined Method for Effective Clustering based on Parallel SOM and Spectral Clustering

Lukáš Vojáček, Jan Martinovič, Kateřina Slaninová,
Pavla Dráždilová, and Jiří Dvorský

Department of Computer Science, FEI, VSB – Technical University of Ostrava,
17. listopadu 15, 708 33, Ostrava-Poruba, Czech Republic
lukas.vojacek@vsb.cz, jan.martinovic@vsb.cz, slaninova@opf.slu.cz,
pavla.drazdilova@vsb.cz, jiri.dvorsky@vsb.cz

Abstract. The paper is oriented to the problem of clustering for large datasets with high-dimensions. We propose a two-phase combined method with regard to high dimensions and exploiting the standard clustering algorithm. The first step of the method is based on the learning phase using artificial neural network, especially Self organizing map, which we find as a suitable method for the reduction of the problem complexity. Due to the fact, that the learning phase of artificial neural networks can be time-consuming operation (especially for large high-dimensional datasets), we decided to accelerate this phase using parallelization to improve the computational efficiency. The second phase of the proposed method is oriented to clustering. Because the visualization provided by Self organizing maps is depending on the map dimension, and is not as clear and comprehensible in the cases of clustering applications, we decided to use spectral clustering algorithm to obtain sufficient clusters. According to our results, the proposed combined method is sufficiently rapid and quite accurate.

1 Theoretical Background

Artificial neural networks (ANN) are the mathematical models inspired by the structure and functionality of the biological neural systems capable of parallel and distributed computation [11]. The ANN can be thought of the non-linear statistical data modeling tool to find and visualize complex relationships between the input data collections and the output map. Moreover, ANN can be used as an adaptive system that is able to change its structure through the learning phase in relation to external input or internal information in the network.

The model typically consists of interconnected groups of neurons, organized in the layers of the system. The basic system of ANN has three layers (the input neurons, the second layer of neurons and the output layer of neurons). All the layers are interconnected through synapses, which have assigned weights used for the calculations of network function $f(x)$. The network function is then defined as a composition of other functions appropriate to the neurons on each layer of the network.

V. Snášel, J. Pokorný, K. Richta (Eds.): Dateso 2011, pp. 120–131, ISBN 978-80-248-2391-1.

ANN is typical of its possibility of learning. Given a class of functions F, learning is a process of finding the optimal solution $f^* \in F$ for a specific task, using a set of observations. For the efficiency measurement is used a cost function $C : F \to \mathbb{R}$ such, that for the optimal solution f^* is $C(f^*) = \min C(f), \forall f \in F$.

There are known two basic approaches to the learning phase of ANN: supervised learning, unsupervised learning and their extensions or combinations. Supervised learning is the approach, where the network function is inferred from the supervised training data collection. The set of training examples consists on pairs of the input vectors and the appropriate output values. The network function is then typically used for pattern recognition or classification (for discrete output) or for regression (for continuous output). As an example of commonly used supervised algorithms, we can mention multilayer Perceptron with Backpropagation method.

Unsupervised learning approach is used for solving the problems oriented to discovery and determination of the data structure. Among commonly used algorithms we can include Self organizing maps (SOM) and its extensions. The ANN with unsupervised learning are commonly used for the tasks like clustering, estimation of statistical distributions, filtering or compression problems.

1.1 Self Organizing Maps

Self organizing map (SOM), also called Kohonen map, is a type of artificial neural network invented by professor Teuvo Kohonen in 1982 [16]. The input space of the training samples is represented in a low dimensional (often two-dimensional) space, called map. The model is capable of projecting the high-dimensional space to the lower-dimensional space [19] and is efficient in the structure visualization due to its feature of the topological preservation using a neighborhood function. The obtained low-dimensional map is often used for pattern detection, clustering, or for characterization and analysis of the input space.

SOM technique has been applied in many spheres like speech recognition [21, 8], image classification [15, 2], document clustering [14, 9] etc. The detailed description of the SOM application is provided in [8].

The model of SOM consists of two layers of nodes. The input layer for receiving and transmitting the input information and the output layer called the map represented the output characteristics. The output layer is commonly organized as the two-dimensional map of nodes, but there are known extensions as, for example, hexagonal grid of output layer. The both layers are feed-forward connected. It is known, that the maps with the smaller grid of the output layer have the behavior similar to K-means clustering [1]. Thr larger output maps have the ability to describe the topological characteristics of the input data collection (often with using U-Matrix for the interpretation of the distance between the nodes).

The SOM input layer is given by input vectors $\boldsymbol{x} \in \mathbb{R}^n$. Each node of this layer is then connected with one of the output nodes K by means of a weight of reference vector $\boldsymbol{w}_k \in \mathbb{R}^n, k = 1, \ldots, K$. During the learning phase, the weight vector $\boldsymbol{w}_k(t)$ is computed for the network at time t, where $t = 0, 1, \ldots$ is discrete time index for each input vector $\boldsymbol{x}(t)$. The passage through the network at time t is an epoch. The learning (training) phase is performed through competitive learning, where for each training example $\boldsymbol{x}(t)$ is computed similarity to all weight vectors $\boldsymbol{w}_k(t)$. The output neuron with the most similar vector is then called as the *best matching unit* (BMU). The weights of the winning neuron and the neurons in the closest neighborhood are then updated and adjusted for the appropriate input vector. The weight vector initialization is commonly assigned randomly, or by using other data mining methods. Concrete implementation of the SOM depends on the method used for the weight vectors' actualization during the training phase.

SOM networks are especially suitable for hidden knowledge presentation. Both the structure of data clusters and query result can be easily visualized. For the overall view of learned data, we use the so-called *Unified distance matrix* (U-matrix), which records the values in clusters and cluster boundaries. The values are assigned to the neuron which wins competition for them, and the distances between neighbouring neurons are recorded with grayness level. Darker colors usually mean greater distance. On the other hand, close data can be colored with similar colors, in this case the boundary between clusters is shown as a steep change in color hue.

1.2 Spectral Clustering

Spectral clustering algorithm uses eigenvalues and eigenvectors of a similarity matrix derived from the data set to find the clusters.

Given a set of data points $\{x_1, \dots x_n\} \in \mathbb{R}^l$ and similarity (cosine measure) $a_{ij} \geq 0$ between all pairs of the data points x_i and x_j.

Let $G = (V, E)$ be an undirected graph with vertex set $V = \{v_1, \dots, v_n\}$. Each vertex v_i in this graph represents the data point x_i. Two vertices are connected, if the similarity a_{ij} between the corresponding data points x_i and x_j is positive, and the edge is weighted by a_{ij}. The weighted adjacency matrix of the graph is the matrix $A = (a_{ij})\ i, j = 1, \dots, n$. If $a_{ij} = 0$ than $(v_i, v_j) \notin E(G)$. For undirected graph it governs that A is symmetric. The degree of a vertex $v_i \in V$ is defined as $d_i = \sum\limits_{j=1}^{n} a_{ij}$. The degree matrix D is defined as the diagonal matrix with the degrees $d_1, \dots, d_n$ on the diagonal. The unnormalized graph Laplacian matrix is defined as $L = D - A$. In[6] Fiedler defines the second smallest eigenvalue $a(G)$ of the of Laplacian matrix $L(G)$ as *algebraic connectivity* of the graph G. In his honor, the corresponding eigenvector is called *Fiedler vector*. The *Spectral Partitioning Algorithm* which uses Fiedler vector is summarized in[22]. The other properties of the algebraic connectivity are in [7].

The survey of data clustering relevant to the clustering document collection is published in [12], while the analysis of spectral clustering is described in [13]. Kannan et al. developed a natural bicriteria measure for assessing the quality of the clustering. How to use the spectral algorithm is studied in [13] by Cheng et al. The practical implementation of the clustering algorithm is presented in [3]. In [5] Ding et al. proposed a new graph partition method based on the min-max clustering principle: the similarity between two subgraphs (cut set) is minimized, while the similarity within each subgraph (summation of similarity between all pairs of nodes within a subgraph) is maximized. Shi and Malik [23] treated image segmentation as a graph partitioning problem and proposed a global criterion, the normalized cut, for segmenting the graph. They showed that an efficient computational technique based on a generalized eigenvalue problem can be used to optimize this criterion. Recursive algorithm is used in [4]. Dasgupta et al. analyzed the second eigenvector technique of spectral partitioning on the planted partition random graph model, by constructing a recursive algorithm.

2 SOM Partitioning

2.1 SOM Algorithms

There are known several variants of the SOM algorithm interpretations [17, 20]. Depending up to the implementation, we can use serial or parallel version of the algorithms.

Serial SOM Algorithms As conventional variant of the serial algorithm interpretations can be considered standard On-line SOM.

On-line SOM Algorithm is the conventional method, where the weight vectors $\boldsymbol{w}_k(t)$ are updated during the training phase recursively for each input vector $\boldsymbol{x}(t)$. The BMU d_c is commonly selected by calculating the similarity using Euclidean distance:

$$d_k(t) = \|\boldsymbol{x}(t) - \boldsymbol{w}_k(t)\|^2, \tag{1}$$

$$d_c(t) \equiv \min_k d_k(t). \tag{2}$$

The weight vectors are then updated using a learning-rate factor $\sigma(t)$ and a neighborhood function $h_{ck}(t)$:

$$\boldsymbol{w}_k(t+1) = \boldsymbol{w}_k(t) + \sigma(t)h_{ck}(t)[\boldsymbol{x}(t) - \boldsymbol{w}_k(t)]. \tag{3}$$

The learning-rate factor $\sigma(t)$ is used for the correction of the weight vectors; during the learning phase is reduced. The concrete updated weight vectors $\boldsymbol{w}_k(t)$ are set by the neighbor function $h_{ck}(t)$, which determines the distance between nodes c and k. The distance is typically decreasing during the learning phase, from an initial value (often comparable to the dimension/or the half of dimension of the lattice) to the value equal to one neuron (one node in the lattice). Commonly is used the standard Gaussian neighborhood function. For the serial online SOM algorithm were published several variants to improve its computational efficiency; as an example we can mention WEBSOM [18].

Parallel SOM Algorithms Till lately, most of the conventional algorithms were designed as sequential. The sequential algorithms were well suited to the past generation of computers, which basically performed the operations in the sequential fashion. With the development of the parallel computation, where the operations were performed simultaneously, there is growing the necessity to redesign the serial algorithms to their parallel implementations.

The parallelization of SOM learning algorithms can be implemented by the network partitioning. The *network partitioning* is the implementation, where the neural network is partitioned among the processors. Then, each input data sample is processed by its assigned processor or the parallel task. The network partitioning was implemented by several authors [10, 24].

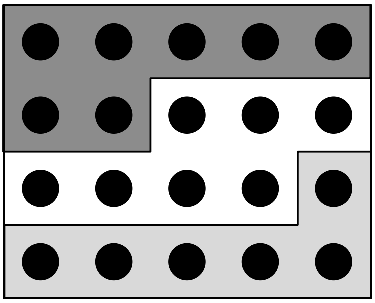


Fig. 1. SOM Map Division

The main principle of our parallel implementation is based on division of the neural network into the parts, where each part is assigned to one processor. This division is shown in the Fig. 1, where we have the map of 5 × 4 nodes. This map is divided into 3 parts which are associated with 3 processors. Not always there is the possibility to divide the map into the identical parts. In these cases, there is the neural network (map) divided using the principle, that the parts of the network differ in at the most one neuron.

The training phase is based on the serial version of the SOM algorithm. Each process finds its own BMU in its part of the map; this node is then compared with other BMU obtained by other processes. The information about the BMU of the whole network is then transmitted to all the processes, which in accordance with this information update weights of the appropriate nodes in their parts of the network.

2.2 Spectral Clustering

The experiment is oriented to the problem of effective clustering for the large datasets with the high-dimension. Due to this reason, we had to find suitable (sufficiently adequate) method for the reduction of the problem complexity. We have decided to used the SOM method. Consider the training set of n-dimensional objects $O = \{o_{ij}; o_i \in \mathbb{R}^n, i = 1, \dots m\}$, where $|O| = m$. SOM allows us to transfer the original problem with $m \times n$ dimension to 2-dimensional matrix A of dimension $l \times l$ represented by the SOM map, where $l \ll \max(m, n)$. Our motivation was to facilitate tasks with objects of high dimension.

Because the visualization provided by SOM is depending on the map dimension, and is not as clear and comprehensible in the cases of clustering applications, we decided to use the appropriate algorithm for clustering. As the SOM map can be thought as a graph, and the node weights can be thought as similarity measures, we decided to use the spectral clustering method for dividing the

SOM map into the clusters with the close nodes. In other words, we transformed the SOM map to the similarity matrix of the individual nodes.

For each graph G, represented by the similarity matrix, the second smallest eigenvalue λ_2 of the Laplacian matrix designates algebraic connectivity $a(G)$ [6]. This value can be represented as a lower bound for edge and vertices graph connectivity [7]. We used this knowledge while computing eigenvector incident to this second smallest eigenvalue (algebraic connectivity).

By the recursive application of minimal cut to the graph we have obtained the sequence of the clusters. For each connected subgraph G_i we have obtained its $a(G_i)$, algebraic connectivity. The graph division is finished, when $a(G_i)$ becomes descending as presented in Algorithm 1.

Algorithm 1 Application of Spectral Clustering Method to SOM

```
1. Construction of the SOM map.
2. Consider graph $G_1 = (V, E)$, which is given by the similarity matrix $A(G_1)$ of dimension $l \times l$ from the SOM map.
3. Construct Laplacian matrix $L(G_1) = D(G_1) - A(G_1), d_{ii} = \sum_{i=1}^{l} a_{ij}$.
4. Compute $a(G_1)$, $u(G_1)$, where algebraic connectivity $a(G_1) = \lambda_2(G_1)$, $u(G_1)$ is appropriate eigenvector.
5. Divide graph $G_j$ to connected subgraphs $G'_{j+1} = \{v_i \in V_j$, where $u_i \leq 0\}$, $G''_{j+1} = \{v_j \in V_j$, where $u_j > 0\}$, but it is possible that this subgraph is not connected. Then find the all connected components which create the subgraphs $G^{(i)}_{j+1}$.
6. For each subgraph compute $a(G^{(i)}_{j+1})$ and appropriate $u(G^{i}_{j+1})$.
7. If $a(G_{j+1}) < \theta$ then do not divide else step 5, where $\theta \in \mathbb{R}$ is the threshold for algebraic connectivity.
```

3 Experiments

The experiments were divided into two phases according to the phases of proposed algorithm. The first phase of the experiments was oriented to the acceleration of the SOM algorithm. As mentioned above, we have tested the parallel implementation of the SOM algorithm training phase for various dimensions of the SOM map and the input vector.

The second phase of the experiments was related to the division of the SOM map using spectral clustering. Both phases are documented by obtained issues and the appropriate figures, see below.

3.1 SOM Acceleration

All the experiments were performed on Windows HPC server 2008 with 6 computing nodes, where each node has 8 processors with 12 GB of memory.

The first experiment was provided on the training set of 300 2-dimensional samples, while the dimension of the neural network (SOM map) was changed. The outputs are presented in the Table 1, where the records with asterisk (*) were provided only by one computing node. In these cases, there is not provided the network communication between processes and due to this fact is the computation faster.

Table 1. Computing Time Dependence on Map Dimension and Number of Processors

Processors	Computing Time [hh:mm:ss]		
	Map Dimension 100 × 100	Map Dimension 300 × 300	Map Dimension 500 × 500
1*	0:01:20	0:12:57	0:35:57
8*	0:00:16	0:01:56	0:21:43
16	0:00:14	0:01:06	0:03:18
24	0:00:13	0:00:50	0:02:05
32	0:00:15	0:00:48	0:01:37
40	0:00:15	0:00:38	0:01:21

The second experiment was provided on the map with selected dimension of 100 × 100 nodes, while the input vector dimension of the training data set was changed. There was used the training set of 150 records. The outputs are presented in the Table 2, where the records with asterisk (*) were provided only by one computing node. In this cases, there is not provided the network communication between processes; due to this fact is the computation faster.

Table 2. Computing Time Dependence on Input Vector Dimension and Number of Processors

Processors	Computing Time [hh:mm:ss]	
	Input Vector Dimension 4	Input Vector Dimension 8
1*	0:06:29	0:12:08
8*	0:01:01	0:01:48
16	0:00:36	0:01:00
24	0:00:27	0:00:44
32	0:00:25	0:00:38
40	0:00:23	0:00:34

As we can see from Tables 1 and 2, the acceleration of the SOM algorithm is appreciable. With growing number of processors is increasing the computation effectiveness, and the computational time is sufficiently reducing.

3.2 SOM Map Division

The next phase of the experiments was oriented to the testing of the proposed algorithms for the SOM map division using spectral clustering, concretely with the Fiedler vector. We have used three training data collections called TwoDiamonds, Lsun a Hepta from the Fundamental Clustering Problems Suite (FCPS)[1]. Short description of selected dataset used in our experiments is given in Table 3.

Table 3. Fundamental Clustering Problems Suite – selected datasets

Name	Cases	#Vars	#Clusters	Main Clustering Problem
Hepta	212	3	7	different densities in clusters
Lsun	400	2	3	different variances in clusters
TwoDiamonds	800	2	2	touching clusters

For comprehensible interpretation we have used U-matrix and its 3D visualization of the SOM map obtained after the dimension reduction of the input dataset . The visualization of the first input data set *TwoDiamonds*, the SOM map after the first phase of proposed algorithm and its division by spectral clustering are presented on the Fig. 2. The U-matrix for this dataset, presented on the Figs. 2(b) and 2(c), is accurately matching to the division provided by the Fiedler vector on the Fig. 2(d).

The same situation occurs for the second input data set *Lsun*, for visualization see Fig. 3(a). As we can see from the Figs. 3(b) and 3(c), the division provided by the SOM and Fiedler vector, Fig. 3(d), is also accurately matching as in the experiment with the first dataset.

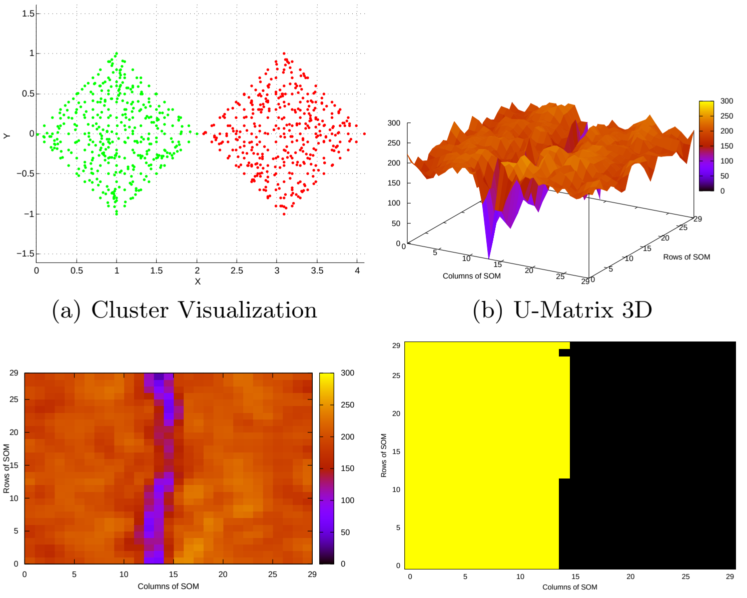
The visualization of the third input data set *Hepta*, the SOM map after the first phase of proposed algorithm and its division by spectral clustering are presented on the Fig. 4. From the Figs. 4(b) and 4(c) we can see, that the division provided by Fig. 4(d) is not as corresponding as in both previous experiments. The experiment with dataset Hepta demonstrates, that some edges are determined in spectral clustering differently then in U-Matrix, but both results are correct.

4 Conclusion

In this paper we presented the parallel implementation of the SOM neural network algorithm. Parallel implementation was tested on HPC cluster containing 6 nodes and 40 processor cores. The achieved speed-up was very good.

Moreover, the partitioning of the resulting SOM map using spectral clustering method was presented. The spectral clustering was applied on U-matrix. This method automatically detected the parts of the SOM that represent the

[1] `http://www.uni-marburg.de/fb12/datenbionik/data?language_sync=1`



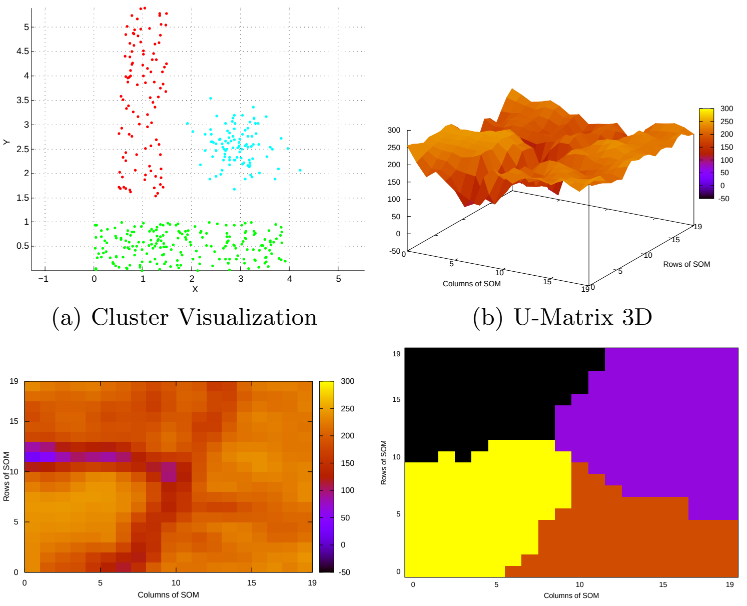
(a) Cluster Visualization

(b) U-Matrix 3D

(c) U-Matrix

(d) SOM Division using Fiedler Vector

Fig. 2. Fundamental Clustering Problems Suite – Two Diamonds



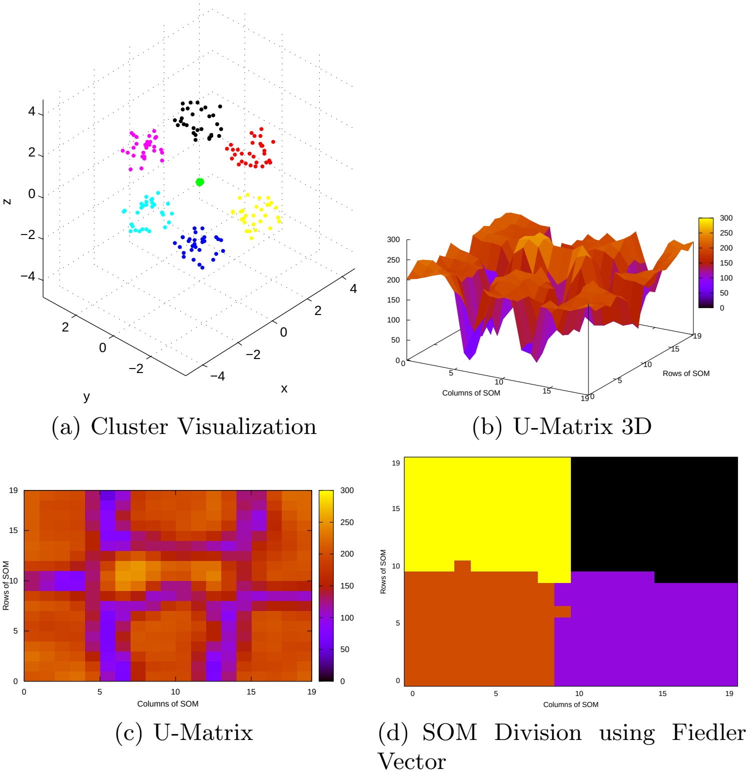
(a) Cluster Visualization

(b) U-Matrix 3D

(c) U-Matrix

(d) SOM Division using Fiedler Vector

Fig. 3. Fundamental Clustering Problems Suite – Lsun



(a) Cluster Visualization

(b) U-Matrix 3D

(c) U-Matrix

(d) SOM Division using Fiedler Vector

Fig. 4. Fundamental Clustering Problems Suite – Hepta

clusters in original high-dimensional data. The detected clusters correspond to the clusters perceived by the human being.

In the future work we intend to focus on more precise specification of the threshold for the algebraic connectivity in the spectral clustering.

Acknowledgment

This work is partially supported by Grant of Grant Agency of Czech Republic No. 205/09/1079, and SGS, VSB – Technical University of Ostrava, Czech Republic, under the grant No. SP2011/172.

References

1. F. Bacao, V. Lobo, and M. Painho. Self-organizing maps as substitutes for k-means clustering. In *Computational Science - ICCS 2005, Pt. 3, Lecture Notes in Computer Science*, pages 209–217, 2005.
2. H. Bekel, G. Heidemann, and H. Ritter. Interactive image data labeling using self-organizing maps in an augmented reality scenario. *Neural Networks*, 18(5-6):566–574, June-July 2005.
3. D. Cheng, R. Kannan, S. Vempala, and G. Wang. On a recursive spectral algorithm for clustering from pairwise similarities. Technical report, MIT, 2003.
4. A. Dasgupta, J. Hopcroft, R. Kannan, and P. Mitra. Spectral clustering by recursive partitioning. In *ESA'06: Proceedings of the 14th conference on Annual European Symposium*, pages 256–267. Springer-Verlag, 2006.
5. C. H. Q. Ding, X. He, H. Zha, M. Gu, and H. D. Simon. A min-max cut algorithm for graph partitioning and data clustering. In *ICDM '01: Proceedings of the 2001 IEEE International Conference on Data Mining*, pages 107–114, Washington, DC, USA, 2001. IEEE Computer Society.
6. M. Fiedler. Algebraic connectivity of graphs. *Czechoslovak Mathematical Journal*, pages 298 – 305, 1973.
7. M. Fiedler. A property of eigenvectors of nonnegative symmetric matrices and its application to graph theory. *Czechoslovak Mathematical Journal*, pages 619–633, 1975.
8. B. Gas, M. Chetouani, J.-L. Zarader, and C. Charbuillet. Predictive kohonen map for speech features extraction. In W. Duch, J. Kacprzyk, E. Oja, and S. Zadrozny, editors, *Artificial Neural Networks: Formal Models and Their Applications - ICANN 2005*, volume 3697 of *Lecture Notes in Computer Science*, pages 793–798. Springer Berlin / Heidelberg, 2005.
9. A. Georgakis and H. Li. An ensemble of som networks for document organization and retrieval. In *Proceedings of AKRR'05, International and Interdisciplinary Conference on Adaptive Knowledge Representation and Reasoning*, pages 141–147, 2005.
10. W. Gropp, E. Lusk, and A. Skjellum. *Using MPI: portable parallel programming with the message-passing inferace.* MIT Press, 1999.
11. S. Haykin. *Neural Networks. A Comprehensive Foundation.* Macmillan, New York, 1994.

12. D. Húsek, J. Pokorný, H. Řezánková, and V. Snášel. Data clustering: From documents to the web. In *Web Data Management Practices: Emerging Techniques and Technologies*, chapter Data Clustering: From Documents to the Web, pages 1–33. Idea Group Inc, 2006.
13. R. Kannan, S. Vempala, and A. Vetta. On clusterings: Good, bad and spectral. *J. ACM*, 51(3):497–515, May 2004.
14. K. Kishida. Techniques of document clustering: A review. *Library and Information Science*, 35(1):106–120, January 2005.
15. O. Kohonen, T. Jaaskelainen, M. Hauta-Kasari, J. Parkkinen, and K. Miyazawa. Organizing spectral image database using self-organizing maps. *Journal of Imaging Science and Technology*, 49(4):431–441, July-August 2005.
16. T. Kohonen. *Self-Organization and Associative Memory*, volume 8 of *Springer Series in Information Sciences.* Springer, Berlin, Heidelberg, 1984. 3rd ed. 1989.
17. T. Kohonen. Things you haven't heard about the self-organizing map. In *Proc. ICNN'93, International Conference on Neural Networks*, pages 1147–1156, Piscataway, NJ, 1993. IEEE, IEEE Service Center.
18. T. Kohonen. Exploration of very large databases by self-organizing maps. In *Proceedings of ICNN'97, International Conference on Neural Networks*, pages PL1–PL6. IEEE Service Center, Piscataway, NJ, 1997.
19. T. Kohonen. *Self Organizing Maps.* Springer-Verlag, 3rd edition, 2001.
20. R. D. Lawrence, G. S. Almasi, and H. E. Rushmeier. A scalable parallel algorithm for self-organizing maps with applications to sparse data mining problems. *Data Mining and Knowledge Discovery*, 3:171–95, 1999.
21. S. Meenakshisundaram, W. L. Woo, and S. S. Dlay. Generalization issues in multiclass classification - new framework using mixture of experts. *Wseas Transactions on Information-Science and Applications*, 4:1676–1681, Dec 2004.
22. A. Pothen, H. D. Simon, and K.-P. Liou. Partitioning sparse matrices with wigenvectors of graphs. *SIAM J. Matrix Anal. Appl.*, 11(3):430–452, July 1990.
23. J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22:888–905, 1997.
24. C. H. Wu, R. E. Hodges, and C. J. Wang. Parallelizing the self-organizing feature map on multiprocessor systems. *Parallel Computing*, 17(6–7):821–832, September 1991.

Analysis of the DBLP Publication Classification Using Concept Lattices

Saleh Alwahaishi, Jan Martinovič, and Václav Snášel, and Miloš Kudělka

Department of Computer Science, FEECS, VŠB – Technical University of Ostrava,
17. listopadu 15, 708 33 Ostrava-Poruba, Czech Republic
{salehw, jan.martinovic, vaclav.snasel}@vsb.cz

Abstract. The definitive classification of scientific journals depends on their aim and scope details. In this paper, we present an approach to facilitate the journals classification of the DBLP datasets. For the analysis, the DBLP data sets were pre-processed by assigning each journal attributes defined by its topics. It is subsequently shown how theory of formal concept analysis can be applied to analyze the relations between journals and the extracted topics from their aims and scopes. It is shown how this approach can be used to facilitate the classifications of scientific journals.

1 Introduction

Formal Concept Analysis (FCA) was invented in the early 1980s by Rudolf Wille as a mathematical theory [1]. FCA is concerned with the formalization of concepts and conceptual thinking and has been applied in many disciplines such as software engineering, knowledge discovery and information retrieved during the last two decades. The mathematical foundation of FCA is described in [2]. In this paper, we describe how we used FCA to create a visual overview of the DBLP scientific journals classification based on their aims and scopes. As a case study, we zoom in on the top journals based on their impact factors.

FCA is a mathematical theory for concepts and concept hierarchies that reflects an understanding of "concept". It explicitly formalizes extension and intension of a concept, their mutual relationships, and the fact that increasing intent implies decreasing extent and vice versa. Based on lattice theory, it allows deriving a concept hierarchy from a given dataset. FCA is thus complementing other conceptual knowledge representations; and the combination of FCA with other representations has been the topic of many publications. For instance, several approaches combined FCA with description logics [3, 4] and with conceptual graphs [5, 6].

The remainder of this paper is composed as follows: In section 2 we introduce an overview of the Digital bibliography and Library project (DBLP). Section 3 visualizes, with an example, the literature using FCA lattice. In section 4 we explained the classification criterion of journals and applied the concept lattice on the selected journals. Section 5 concludes the paper.

V. Snášel, J. Pokorný, K. Richta (Eds.): Dateso 2011, pp. 132–139, ISBN 978-80-248-2391-1.

2 Digital Bibliography & Library Project (DBLP)

Digital libraries are collections of resources and services stored in digital formats and accessed by computers. Studying them offers an interesting case study for researches for the following reasons: Firstly, they grow quickly; secondly, they represent a multidisciplinary domain which has attracted researchers from a wide area of expertise. 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 one million articles on computer science in January 2010. These articles were published in Journals such as VLDB, the IEEE and the ACM Transactions and Conference proceedings [7, 8]. Besides 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.

In addition to using the DBLP dataset for finding academic experts, it has been used extensively in academic recommender systems. A number of studies were conducted to recommend academic events and collaborators for researchers using different methods and techniques. For example, a recommender system for academic collaboration called DBconnect was presented in [9]. Authors of this paper used DBLP data to generate bipartite (author-conference) and tripartite (authorconference-topics) graph models, and designed a random walk algorithm for these models to calculate the relevance score between authors. And in another study [10] a recommender system for events and scientific communities for researchers was proposed based on social network analysis.

Querying large datasets produces large sets too, which makes the user unable to decide from where he has to start looking at the results. To solve this problem clustering and ranking were suggested in many papers. A system to visualize author information and relationships simultaneously was presented in [11]. The authors applied two types of clustering, keyword clustering and author clustering to visualize the relationships and groupings of authors. In [12] document clustering was applied to provide an overview of the recent trends in data mining activities. Clustering and ranking are often applied separately but in [13] a novel framework called RankClus was proposed to integrate them. To increase the accuracy of IR clustering, the authors in [14] proposed transferring knowledge available on the word side to the document side; they introduced a model based on nonnegative matrix factorization to achieve it.

3 Concept Analysis Of Journals Classification

This section describes how formal concept analysis is employed to analyze the DBLP's journals classification. Formal Concept Analysis can be used as an unsupervised clustering technique. The starting point of the analysis is a database table consisting of rows *G* (i.e. objects), columns *M* (i.e. attributes) and crosses *I*

$\subseteq G{\times}M$ (i.e. relationships between objects and attributes). The mathematical structure used to reference such a cross table is called a formal context (*G, M, I*).

A group of interested similar journals, which covered the scope of computer science, were selected. The list of selected journals (objects) was obtained from well-known DBLP database that contains information about the published articles and their authors as well. The selected list of links to journals has the size of 115 items. The next step was to identify main topics (attributes), which each of the journals covers. From the journal web sites we have found the aim and scope of each journal, and have manually extracted the main topics, such as Pattern Recognition, Image Processing, etc. Each journal has been identified by an existing classifier by company due to the problem with using their own names or similar names of topics. The used classifier that contains about 1224 sub disciplines classified to disciplines and those classified to discipline field, e.g. sub discipline Pattern Recognition is in disciplines Artificial Intelligence and Image Processing and that is in Information and computing sciences [15]. We selected only sub disciplines in the field Technology and Information and computing sciences. Our manually extracted topic from journals in many cases correspond the classified disciplines, but in some cases it was necessary to assign the extracted topic to sub discipline, which was almost similar. Therefore, journals were classified into a list of topics based in their relation to the topic. The classification process ends up with ten main topics that have twenty nine subfields or disciplines. Table 2 shows the main topics and their subfields.

A journal is represented as a list of topics. The topics are the disciplines that being covered by all journals, based on the extracted data from their aims and scopes. Each topic is assigned a weight of 0 or 1. A topic's weight for a journal expresses the coverage possibility of the topic by the related journal. A value of 1 denotes that the journal covers the column's topic and 0 denotes the lack of coverage. Formally, these data can be represented as a matrix of journals by topics whose *m* rows and *n* columns correspond to *m* journals and *n* topics, respectively. The elements of the journal-topic matrix are the weights of each term for a particular document, that is:

$$\boldsymbol{Y} = \begin{bmatrix} y_{11} & \cdots & y_{1n} \\ \vdots & & \vdots \\ y_{m1} & \cdots & y_{mn} \end{bmatrix}$$

Where y_{ij} denotes the weight assigned to topic T_j for journal J_i.

The formal concept analysis of the data starts with the creation of a formal context. The formal objects of the formal context are the journals J_i that were retrieved from DBLP database. The set of these journals is denoted by *J*. Using the information that was extracted from the aim and scope of the journals in *J*. The coverage possibility T_j that shows the topic coverage by the journals in *J*, constitute the formal attributes of the formal context. The set containing these attributes is denoted by T.

The cross table of the resulting formal context has a row for each journals in *J*, a column for each topic in *T* and a cross in the row of J_i and the column of T_j if the corresponding weight y_{ij} is 1. To minimize the cross table size, journals impact factors will be considered to decrease the number of tested journals. The journals with an impact factor of 3.0 and above will be enlisted in the matrix, dropping the number of selected journals to be 18 as shown in Table 3. After the formal context is constructed, formal concept analysis is applied to produce the concept lattice.

Table 4 represents the formal context. A cross in the row of J_i and the column of T_j indicates that T_j is believed to be a covered topic by the journal of J_i.

Table 1. Journals' impact factors and abbreviations

Abbreviation	Journal	Impact Factor
A	Nucleic Acids Research	6.878
B	IEEE Transactions on Pattern Analysis and Machine Intelligence	5.96
C	International Journal of Computer Vision	5.358
D	Computer Applications in the Biosciences	4.328
E	Journal of Selected Areas in Communications	4.249
F	Transactions on Medical Imaging	4.004
G	Transactions on Information Theory	3.793
H	BMC Bioinformatics	3.78
I	Transactions on Neural Networks	3.726
J	Journal of Chemical Information and Computer Sciences	3.643
K	Transactions on Fuzzy Systems	3.624
L	Journal of Computational Chemistry	3.39
M	Transactions on Graphics	3.383
N	Transactions on Mobile Computing	3.352
O	Transactions on Image Processing	3.315
P	Pattern Recognition	3.279
Q	Automatica	3.178
R	Information Sciences	3.095

The intent of each formal concept contains precisely those topics covered by all journals in the extent. Conversely, the extent contains precisely those journals sharing all topics in the intent.

The line diagram of the concept lattice, showing the partially ordered set of concepts is shown in Fig 1, has the minimal set of edges necessary; all other edges can be derived by using reflexivity and transitivity. Journals and topics label the node that represents the formal concept they generate. All concept nodes above a node labeled by a journal have the journal in their extent. All concept nodes below a node labeled by a topic have the topic in their intent. The extent of the concept node labeled by the topic "STVV" for example is easily found by collecting the journal H labeling this concept node on a path going downward.

Table 2. Formal context

	AIIP	**CTM**	**CS**	**ISLIS**	**DF**	**DC**	**CT**	**CA**	**DIP**	**STVV**
A		x	x							
B	x							x		
C	x			x						
D		x	x							
E							x			
F	x	x	x					x		
G	x	x		x	x		x			
H		x	x							x
I			x							
J	x	x		x						
K	x									
L		x	x							
M	x									
N			x			x	x			
O	x				x					
P	x									
Q	x	x	x	x				x	x	
R	x	x	x		x		x	x		

The intent of this concept is found by first collecting the topic “STVV” and by going upward to collect the topic “CTM”, and “CS” labeling the two concepts found on paths going upward. The resulting extent-intent pair of this concept is ({H}, {CTM,CS,STVV}).

The concept generated by the topic “DF” is a sub concept of the concept generated by the topic “AIIP”, for the extent of the former concept is contained in the extent of the latter concept. All journals classified by the topic “DF” were also classified by the topic “AIIP”, suggesting that within the given formal context “DF” is a more specific topic than “AIIP”.

Another multi constructed example is found in the extent of the concept node labeled by the topic “DIP”, which is found by collecting the journal Q labeling this concept node on a path going downward. The intent of this concept is found by collecting the topics “CTM”, “CA”, and “ISLIS” labeling the three concepts found on paths going upward. The latter two topics, however, are sub concepts of the concept generated by the topic “AIIP”. The resulting extent-intent pair of this concept is ({Q}, {AIIP,CTM,CS,ISLIS,CA,DIP}).

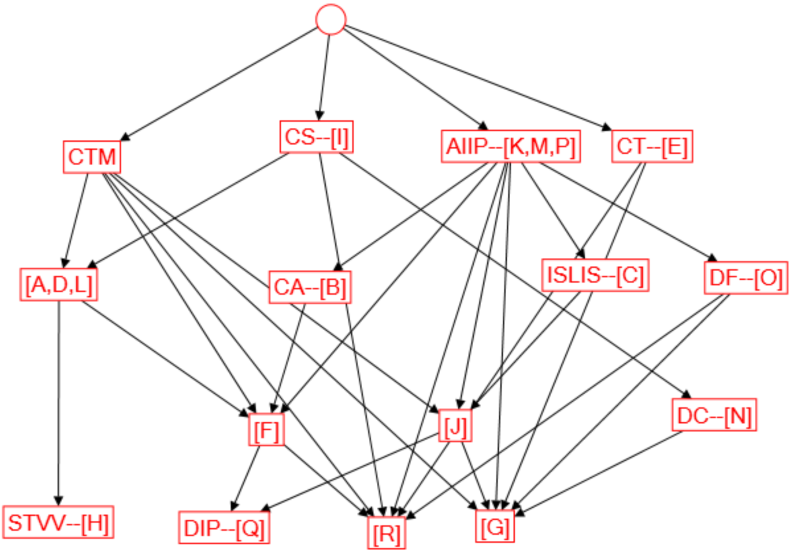


Fig. 1. Concept lattice for journals classification

4 Conclusion

The concept lattice uncovers relational and contextual information. Journals' topic categorizations are put into relational context depending on how they are associated by the journals' aims and scopes. The topics "Computer Theory and Mathematics – CTM", and "Data and Information Processing –DIP" for example are shown as related because these topics share a similar classification context. The implicit structures revealed help researchers to classify journals more efficiently. This approach has the potential to support the emergence of new knowledge by identifying concept relations, making these explicit and enabling researchers to inspect these concept relations.

Concept lattices are not intended to build or substitute traditional static ontologies, rather they aim to support specifications of less rigorous relations, or associations [16], which might be more intuitive to knowledge workers and lead to more interesting links via associations.

Abbreviation	Main Topic	Subfields
AIIP	Artificial Intelligence and Image Processing	Adaptive Agents and Intelligent Robotics, Neural, Evolutionary and Fuzzy Computation, Simulation and Modeling, Computer Vision Pattern Recognition and Data Mining, Signal processing, Image Processing
CTM	Computation Theory and Mathematics	Computer Graphics
		Other Computation Theory and Mathematics
		Numerical Computation
		Applied Discrete Mathematics
		Computational Logic and Formal Languages
		Analysis of Algorithms and Complexity
CS	Computer Software	Software Engineering
		Operating Systems
		Computer System Security
		Bioinformatics Software
ISLIS	Information Systems and Library and Information Studies	Database and Database Management
		Information Retrieval and Web Search
		Inter-organizational Information Systems and Web Services
		Information Systems Management
		Information Systems Development Methodologies
DF	Data Format	Data Encryption
		Data Structures
DC	Distributed Computing	Mobile Technologies
		Distributed Computing
CT	Communications Technologies	Computer Communications Networks (computer network)
		Wireless Communications
		Other Communications Technologies (telecommunications)
CA	Computer Architecture	
DIP	Data and Information Processing	
STVV	Software Testing and Verification & Validation	

Table 3. Journals’ main topics and subfields

5 References

1. Wille. R. (1982). Restructuring lattice theory: an approach based on hierarchies of concepts. In I. Rival (Ed.). Ordered sets. Reidel. Dordrecht-Boston. 445-470.
2. Ganter, B., Wille, R. (1999) Formal Concept Analysis: Mathematical foundations. Springer
3. Beydoun, G. (2009) Using Formal Concept Analysis towards Cooperative E-Learning. D. Richards and B.H. Kang (Eds.): PKAW, LNAI 5465, 109-117. Springer
4. Priss, U. (2006), Formal Concept Analysis in Information Science. Cronin. Blaise (ed.). Annual Review of Information Science and Technology, ASIST, Vol. 40.
5. Ganter, B., Kuznetsov, S.O. (2008) Scale Coarsening as Feature Selection. Medina and S. Obiedkov (Eds.) : ICFCA, LNAI 4933, 217-228. Springer.
6. Stumme, G., Wille, R., Wille, U. (1998) Conceptual knowledge discovery in databases using Formal Concept Analysis Methods. PKDD, 450-458.
7. Ley, M. (2002) The dblp computer science bibliography: Evolution, research issues, perspectives. SPIRE 2002: Proceedings of the 9th International Symposium on String Processing and Information Retrieval. London, UK: Springer-Verlag, pp. 1–10.
8. URL, http://en.wikipedia.org/wiki/DBLP.
9. Zaiane ,O. R. Chen, J. and Goebel, R. (2007) Dbconnect: mining research community on dblp data. WebKDD/SNA-KDD '07: Proceedings of the 9th WebKDD and 1st SNA-KDD 2007 workshop on Web mining and social network analysis. New York, NY, USA: ACM, pp. 74–81.
10. R. Klamma, P. M. Cuong, and Y. Cao (2009) You never walk alone: Recommending academic events based on social network analysis. Complex (1), pp. 657–670.
11. Chan, S. Pon, R. and Cardenas, A. (2006) Visualization and clustering of author social networks. Distributed Multimedia Systems Conference, pp. 174–180.
12. Peng, Y. Kou, G. and Shi, Y. (2006) Recent trends in data mining: Document clustering of dm publications. International Conference on Service Systems and Service Management, vol. 2, pp. 1653–1659.
13. Y. Sun, J. Han, P. Zhao, Z. Yin, H. Cheng, and T. Wu, "Rankclus: integrating clustering with ranking for heterogeneous information network analysis," in EDBT '09: Proceedings of the 12th International Conference on Extending Database Technology. New York, NY, USA: ACM, 2009, pp. 565–576.
14. T. Li, C. Ding, Y. Zhang, and B. Shao, "Knowledge transformation from word space to document space," in SIGIR '08: Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval. New York, NY, USA: ACM, 2008, pp. 187–194.
15. Obadi G., Drazdilova P., Hlavacek L., Martinovic J., and Snasel V. (2010) A Tolerance Rough Set Based Overlapping Clustering for the DBLP Data, Web Intelligence and Intelligent Agent Technology, IEEE/WIC/ACM International Conference on, pp. 57-60, 2010 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology.
16. Krohn U, Davies NJ, Weeks, R. (1999) Concept lattices for knowledge management. BT Technology Journal, 17(4):108-116.

Automatic Keyphrase Extraction based on NLP and Statistical Methods

Martin Dostal and Karel Ježek

Department of Computer Science and Engineering, Faculty of Applied Sciences
University of West Bohemia, Plzeň, Czech Republic
{madostal, jezek_ka}@kiv.zcu.cz

Abstract. In this article we would like to present our experimental approach to automatic keyphrase extraction based on statistical methods and Wordnet-based pattern evaluation. Automatic keyphrases are important for automatic tagging and clustering because manually assigned keyphrases are not sufficient in most cases. Keyphrase candidates are extracted in a new way derived from a combination of graph methods (TextRank) and statistical methods (TF*IDF). Keyword candidates are merged with named entities and stop words according to NL POS (Part Of a Speech) patterns. Automatic keyphrases are generated as TF*IDF weighted unigrams. Keyphrases describe the main ideas of documents in a human-readable way. Evaluation of this approach is presented in articles extracted from News web sites. Each article contains manually assigned topics/categories which are used for keyword evaluation.

Keywords: keyphrase extraction, Wordnet, TextRank, TFIDF, NLP

1 Introduction

In this paper we would like to present our experimental approach to automatic keywords and keyphrase extraction. Our approach is very useful in cases where we don't have manual keywords assigned by the author or where these keywords are not sufficient. Our approach builds on ideas from TextRank [1] and RAKE [2] with a combination of statistical (TF*IDF) and NLP methods.

Keywords are defined as a sequence of one or more words and provide a compact description of a document's content. Keywords are often used to define queries within information retrieval systems because they are easy to define, remember, and share. Keywords are usually corpus independent and can be applied across multiple different systems. For example, the Phrasier [3] system lists documents related to a primary document's keywords and supports keyword anchors as hyperlinks between documents. The Keyphind system [4] uses keywords as the basic building block for an IR system especially for summarization and clustering tasks. Hulth [5] (2004) describes Keegle, a system that provides extracted keywords for web pages found by a Google search engine.

Keyphrases consist of two or more keywords and named entities. In our approach, keyphrases consist of keywords, named entities and stop words. Stop words can be

V. Snášel, J. Pokorný, K. Richta (Eds.): Dateso 2011, pp. 140–145, ISBN 978-80-248-2391-1.

an important part of a keyphrase, which increase the readability and intelligibility of a phrase in natural language. This idea was inspired by the RAKE system for automatic keyword extraction from individual documents.

2 Related graph algorithms

In this section, we would like to discuss graph-based ranking algorithms, especially Google's PageRank [6] and its implementation for text document processing.

Google's PageRank [6] is the first and most popular graph-based ranking algorithm which has been successfully used by social networks, citation analysis, and link-structure analysis of the World Wide Web. This algorithm is a way of deciding on the importance of a node within a graph. The importance of a node is evaluated based on global information recursively drawn from the graph. The graph node is important when it is often recommended by other nodes. In this special case, if other web documents contain links in the form of URI to this one.

The TextRank graph-based algorithm is a ranking model for graphs extracted from text documents. The text is split into tokens which represent the nodes of the graph. Nodes are connected with weighted edges based on the lexical or semantic relations, for example. TextRank algorithms use a co-occurrence relation controlled by the distance between word occurrences within a sliding window of maximum N words. The best results were achieved for a maximum of two words which correspond with N-gram based algorithms.

3 Algorithm for automatic keyword extraction

In this section, we would like to discuss our approach to automatic keyword extraction for documents. TextRank can be used in individual documents without any other knowledge. Graph nodes ranking is made upon co-occurrences of tokens and the score of the other nodes with edges to the current one. In our algorithm, the TF*IDF score is used for better text token evaluation instead of the measure based only on n-grams. The next idea is based on two versions of the keyword characteristic. The keyword extraction problem can be divided into:

- Individual keyword extraction – individual words with a special and important meaning, generally in the form of a noun or named entity.
- Keyphrase extraction and derivation – phrases contain two or more keywords and other information in a human-readable form. This phrase can contain verbs and stop words for better readability. Short phrases can be joined by their co-occurrence percentage number.

Our algorithm can be divided into three main steps:

1. Text preprocessing
2. Keyword extraction

3. Keyphrase extraction

During the text preprocessing phase, the article's content is divided into tokens and non-significant characters are removed. Named entities are recognized by the elementary method: the first upper-case letter with corpus-based statistic is good enough for this recognition. The tokens are divided by their POS tag and generally only nouns and adjectives can be declared as potential keywords. The next idea is to choose only common words instead of unique ones. Keywords are often used for clustering and a keyword is useless when it is assigned to only a few articles. This is the reason why we remove tokens with low frequency and set up rules for general nouns. There is no such rule for named entities. The remaining tokens and named entities are declared as keywords candidates. We calculate the TF*IDF score only for these keyword candidates.

The keyword extraction phase contains only the method for removing useless candidates whose TF*IDF score is lower than 1/5 of a maximal value. This boundary can be changed by the number of requested automatic keywords.

The keyphrase extraction part can be described by these steps:

- NLP method – interesting n-grams are chosen. The choice is based on their POS tag patterns and the corpus frequency is counted only for these n-grams. These n-grams can be marked as keyphrase candidates.
- A score of importance is counted for each keyphrase candidate. This score contains the n-gram corpus frequency and TF*IDF score for each word. The score is used for document keyphrase selection.
- Derivation – keyphrase candidates are merged with named entities or individual keywords if their co-occurrence is significant for this document.

Used POS patterns:

A) POS patterns for 3-grams:
   - (N or named entity), (V or A or stop word), (N or named entity)
   - 3x (named entity)

     example: Tim Berners Lee

B) POS patterns for 2-grams
   - A, (N or named entity)
   - 2x (named entity)

     example: Bill Gates

Used tags:

- N – noun,
- V – verb,
- A – adjective,
- “stop word“ – a word from stop list,
- “named entity” – a potential named entity based on simple patterns like word in the middle of the sentence with first capital letter.

Keywords and keyphrases are ordered by their TF*IDF score and only the most important keywords and keyphrases are used. The number of used items naturally depends on the requested number of these items. This number can be chosen directly, or by percentage measure from the score of the best item.

4 Evaluation

To evaluate performance, we tested our system against a collection of newspaper articles about technology that were extracted from the Web. These articles contain manually assigned keywords from a controlled dictionary. These keywords were chosen by the author of the article. Such keywords are marked as topics in our case. These topics cover the article content very sketchily and capture only the basic idea of the article. For example, the content of the article contains the word “Google”, but the manually assigned topic was “Google Inc”. This can be enough for article classification, but for automatic keyword evaluation it is a further challenge.

We have decided that our news corpus is a collection from the “real world” and if the results are satisfying, it would be usable for other general data collections. Another reason was that this approach is very experimental and we had no better article collection for evaluation. Initially, we thought that these results would not be satisfactory, but we were surprised by their high relevance. The size of the data sets was chosen based on the number of the articles that were available for the each subset evaluation.
We have used three data sets:

A) Corpus of 500 articles with a small number of manually assigned topics (600 different topics) by the author.
B) Corpus of 50 random articles with a small number of manually assigned topics. Each article contains approximately 3 manual topics.
C) Corpus of 50 random articles with manually assigned topics (by author) and expanded, by 2-3 another human annotators, to other important topics covered by the article. Each article contains approximately 5 manual topics.

The statistics of precision and recall for part A) are shown in Table 1. These values are calculated for a different boundary configuration of accepted keywords. This threshold is set as the percentage difference from the score of the most important keyword. Precision and recall are reduced because of topic classification difficulties. There are about 600 various topics in this data set and some of them are very similar for the human annotator, but not for our topic classification module.

The topic classification is done only based on the name of the topic, so for example: topics “Google Inc.” and “Google operating system” have the same score for these automatic keywords: “Google” and “Android”.

Table 1. Precision and recall for the corpus of 500 articles.

Boundary	**1%**	**10%**	**30%**	**50%**	**70%**	**90%**
Precision	13.2%	18.9%	27%	31.8%	38.2%	40.8%
Recall	46.1%	33%	22.6%	16.5%	12.8%	10.53%

The statistics of precision and recall for part B) are shown in Table 2. This corpus contains 50 random original articles. The main difference between evaluation A) and B) is in the number of classification topics. There are about 120 topics used for classification so the precision and recall are not distorted as much as in part A).

Table 2. Precision and recall for the corpus of 50 articles.

Boundary	**1%**	**10%**	**20%**	**30%**	**40%**	**50%**	**70%**	**90%**
Precision	30%	38.6%	42.4%	48%	50%	49.4%	50.7%	55.2%
Recall	49%	33%	26.9%	23.7%	22%	18.5%	13.6%	12.9%

The statistics of precision and recall for part C) are shown in Table 3. This corpus contains 50 articles with 2-3 additional human-annotated topics. The total number of manually added topics is about 5.

Table 3. The corpus of 50 articles with additional human annotations.

Boundary	**1%**	**10%**	**20%**	**30%**	**40%**	**50%**	**70%**	**90%**
Precision	37.4%	47.4%	53.9%	55.8%	59.12%	59.4%	60.6%	64%
Recall	54.6%	35.9%	29.3%	23.7%	22.4%	18.8%	14.1%	13.5%

5 Conclusion and future work

The proposed approach seems to be efficient enough to be comparable with other automatic keyword extraction systems. For example, the RAKE system achieved 33.7% precision with 41.5% recall and the undirected TextRank achieved 31.2% precision with 43.1% recall. Our approach achieved 37.4% precision and 54.6% recall for a small corpus with expanded number of annotations, including the problem of keyword generation and automatic clustering. We can assume that precision and recall will be a little bit lower for a bigger corpus. The most significant feature of the corpus is the number of exact manual annotations which are used for performance tests.

In the future, we would like to compare our approach with other methods on their data corpuses. These corpuses were not available at this moment so we had to use our data collection for the first evaluation tests. Automatic keywords will be used for mapping the Linked Data topics to the articles and graph-based knowledge extraction.

References

1. Mihalcea R. and Tarau P. Textrank: Bringing order into texts. In *Proceedings of EMNLP 2004* (ed. Lin D and Wu D), pp. 404–411. Association for Computational Linguistics, Barcelona, Spain.
2. Rose S., Engel D., Cramer N., Cowley D. *Automatic keyword extraction from individual documents.* In Text mining applications and theory 2010, pp. 3-19.
3. Jones S. and Paynter G. Automatic extraction of document keyphrases for use in digital libraries: evaluation and applications. *Journal of the American Society for Information Science and Technology.*
4. Gutwin C, Paynter G, Witten I, Nevill-Manning C and Frank E. *Improving browsing in digital libraries with keyphrase indexes.* Decision Support Systems 27(1–2), 81–104, 1999.
5. Hulth A. *Combining machine learning and natural language processing for automatic keyword extraction.* Stockholm University, Faculty of Social Sciences, Department of Computer and Systems Sciences (together with KTH), 2004.
6. Page, Lawrence and Brin, Sergey and Motwani, Rajeev and Winograd, Terry. *The PageRank Citation Ranking: Bringing Order to the Web.* Technical Report. Stanford InfoLab, 1999.

Flexible Cache for Database Management Systems

Radim Bača and David Bednář

Department of Computer Science, VŠB – Technical University of Ostrava
17. listopadu 15, 708 33 Ostrava-Poruba, Czech Republic
{radim.baca,david.bednar}@vsb.cz

Abstract. Cache of a persistent data structure represents an important part, which significantly influence its efficiency. Cache consists from an array of main memory blocks (caled cache nodes) with a constant size. Cache nodes buffer the data structure nodes hence they can be accessed quickly. However, data structure nodes do not usually fully utilize the whole main memory block. Therefore, the constant cache node size the waste of the main memory. In this article, we propose the solution where the cache consists from the cache nodes with a different size. Cache is split into several cache rows. The data structure nodes with the similar size are stored in one cache row. Such a solution has two problems: (1) that the some cache row has to replace the nodes too often during the querying (cache row contention), and (2) that the data structure nodes has to be moved between the cache rows during the insert and the delete operation. In our experimental section we show that the effect of the cache row contention is minimized if we set the cache row size according to the data structure node distribution. We also discuss a solution of the node cache moves problem. **Key words:** Cache utilization, flexible cache

1 Introduction

The cache of a persistent data structure represents an important part, which significantly influence its efficiency. It consists from an array of main memory blocks called *cache nodes* with a constant size. Cache nodes buffer the *data structure nodes* hence they can be accessed quickly. Note the difference between the cache node and the data structure node which is a logical node of a data structure and it does not have to be loaded in the cache. Major research effort focus on a cache *replacement algorithm* [3, 1, 2], which selects the node that should be removed from the cache array to create free space in the cache for new data structure nodes.

The cache is usually shared by many data structures which can have a different node size in a main memory. Another issue is that data structure node usually does not utilize the whole node. Therefore, the constant cache node size is waste of the main memory since the size of the cache node has to be equal to a largest data structure node in the database system. Similar situation occurs when we use compressed data structures [4] which compress the node when it is stored on a secondary storage, but the node is decompressed in the main memory. Usage of compressed data structures highlight the problem of variable size of a data structure nodes, since two nodes can have a significantly different main memory size even though their compressed size is similar.

In this article we propose the solution called flexible cache containing cache nodes with a different size. Flexible cache is split into several cache rows. Nodes with the similar size are stored in one cache row. It can happen that we have to replace nodes from one cache row more often that in other cache rows. We call this problem *cache row contention*. We size the cache rows according to the node distribution in order to minimize the impact of this problem and to utilize the advantages of the flexible cache. Another problem called the *node cache moves* occurs during the insert and the delete operation. As we insert items into a data structure node its utilization grows. When it becomes too large then we have to move it into another cache row with the bigger cache nodes.

In Section 2, we describe different cache strategies. Section 3 introduces basic ideas of our flexible cache strategy and solutions of its problems. In Section 4, we describe our experimental results and in Section 5 we depict possible future improvements of the flexible cache.

2 Cache of Persistent Data Structures

The databases systems retain the data structure pages in cache nodes for a period of time after they have been read in from the disk and accessed by a particular application. The main purpose is to keep popular nodes in memory and reduce the disk I/O.

There is a number of different cache strategies which are usually used in the persistent data structures. Data is stored in *blocks* on a secondary storage. Block has a constant size varying typically from 2kB to 16kB. Cache contains a set of cache nodes which can be used in order to accommodate data from the block. When a data structure needs a data from some block then it is first loaded into some selected cache node. Selected cache node usually corresponds to a different data structure node, therefore, we have to write the node into a secondary storage first (if the node contains any changes) before we load the new node there. Selection of an appropriate cache node is usually called the *replacement algorithm*. Basically, the major issue is to find a cache node which will not be needed for a longest period of time. Since this kind of prediction is usually not available, there is number of different algorithms commonly used:

- RR (Random replacement) - this algorithm select the node randomly when necessary. This algorithm does not require keeping any information about the access history of nodes.
- LRU (Least Recently Used) - there is a time-stamp assigned to every node. We actualize the time-stamp each time we access the node. This selection algorithm simply select the node with lower time-stamp. This is a basic algorithm with many different variants.
- LFU (Least Frequency Used) - algorithm counts number of node usages. Nodes used least often are discarded first. If the frequency of usage of each node is the same, then nodes are expired by the LRU algorithm.

There is a number of replacement algorithms which usually combines the LRU and LFU [3, 1, 2]. LRU-K algorithm [3] remembers k recent accesses to the node and selects the node considering all these accesses. It is a generalization of the LRU algorithm which is considered as a LRU-1. In the next, we discuss some standard methods.

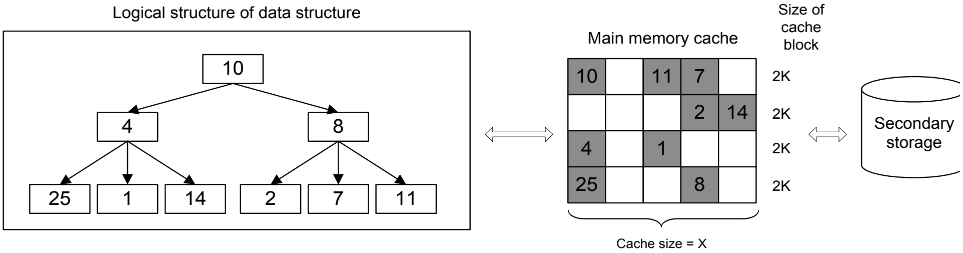


Fig. 1. Cache between physical storage and logical structure - cache size is setup on some constant value of blocks (X). Each block has the same own size, in this case 2K

2.1 LRU-K Page Replacement Algorithm

The basic idea of LRU-K ($k \geq 2$) method is to keep track of the times of the last k references to popular nodes, using the information to statistically estimate the interarrival times of references on a node by node basis. The LRU-K algorithm has many salient properties. It differentiates between the nodes with different levels of reference frequency, it detects locality of reference within query executions across multiple queries in the same transaction or in a multiuser environment. It is self-reliant (not need any external hints) and fairly simple. LRU-K keeps in the memory only those nodes that seem to have an interarrival time to justify their residence. That residence can be nodes with the shortest access interarrival time or equivalently greatest probability of reference, e.g. LRU-1 ($k = 1$) algorithm keeps in the memory only those nodes that seem to have the shortest interarrival time. The algorithm is based on the two data structures:

- K most recent references to nodes n from the history point of view - $HIST(n, x)$, where x represents the time of the last (the second to the last and so on) reference ($x \in 1, ...k$).
- Denotes the time of the most recent reference to node n - $LAST(n)$.

The LRU-2 algorithm provides essentially optimal buffering behavior based on the information given. It has significant performance advantages over conventional algorithm like LRU-1, because discriminate better between frequently referenced and infrequently referenced nodes.

3 Flexible Cache

Here we discuss the ideas behind the flexible cache which allows us to utilize the cache more efficiently.

The nodes of data structures can have a different node utilization, however, database management systems use cache nodes with constant size which is equal to the block size. This leads to a situation that main memory is wasted. In the flexible cache we solve this problem by creating the cache nodes with a different size. This reorganization allows to us store more cache nodes in the main memory which consequently reduces the number of disk accesses which are very time consuming.

Cache is split into several *cache rows* r, where the cache nodes belonging to the same cache row has the equal size *nodesize(r)*. Each cache row r has its own size *count(r)* which represents the number of cache nodes belonging to r. There is a set of cache rows r_1, ..., r_m, where *nodesize(r_i)* $>$ *nodesize(r_{i+1})*. Every data structure node n has its *realsize(n)* which specifies what is the minimal size of main memory where the node in the current state can be stored. The *realsize(n)* is mainly driven by the node utilization.

Data structure node n belongs to the the row r_i:

- If *realsize(n)* $<$ *nodesize(r_i)* and *realsize(n)* $>$ *nodesize(r_{i-1})*, or,
- if i = 1 and *realsize(n)* $<$ *nodesize(r_i)*.

We have to slightly adapt the replacement algorithm for our flexible cache. The node n which should be loaded into the cache belongs to a cache row r_n, therefore, the replacement algorithm search for a appropriate cache node only in row r_n.

Figure 2 shows us the idea on the example. We have the cache with four cache rows. The first row r has cache nodes with *nodesize(r)* equal to the size of a biggest data structure node. It is for the data structure nodes which are almost fully utilized. Other cache rows store smaller data structure nodes, where the smaller node size can be caused by many different parameters (node utilization, bad compression ratio in a case of compressed data structures, data structure with smaller nodes).

Flexible cache has two problems: (1) we could read more often from one cache row then from the other one (cache row contention), and (2) we may be forced to move the data structure node from one cache row to another during the inserts. Both problems has a solution which minimize their influence. We introduces solutions for both problems is Section 3.1 and in Section 5.

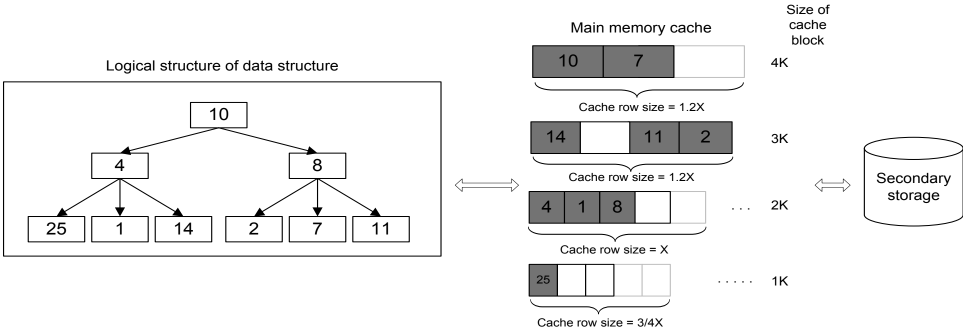


Fig. 2. Cache blocks with different size - cache is split into row with different size of blocks in every row - e.g. 4K to 1K. In the first cache row are stored nodes with the size 4K. Moreover, the rows have no constant and same size itself. The size of each row depends on percentage representation nodes with that size in the persistent structure.

3.1 Cache Row Contention and Node Cache Moves

If the *count(r)* some row r is too small and we replace nodes in this row too often (when compared to other cache rows) then we talk about the cache row contention. The problem can be minimized if we collect the statistics about the node distribution and cache rows are sized according to this distribution. As we will see in experiments, row contention was minimized and the flexible cache performed usually better then the regular cache.

The second problem could be minimized if we have a low priority thread in our database system which moves the data structure nodes from one cache row into the another one when the cache node becomes almost full. Each cache row r_i has a *threshold(*r_i*)* which is a value slightly lower than *nodesize(*r_i*)*. We keep the *move list* of nodes and the data structure node n is added to the list when its size exceed the specified *threshold(*r_i*)*. Thread then move the nodes from the list from one cache row into the another.

4 Experimental results

In our experiments[3], we use two-dimensional real data collection called TIGER (Topologically Integrated Geographic Encoding and Referencing system) containing about 5.5 millions records.

During tests we measure query time (range query randomly created), insert time (time of the creation the data structure) and Disk Access Cost (DAC). The flexible cache solution is compared with the common cache. The code is implemented in C++. Persistent data structure is in our experiments represented by R*-tree and compressed R*-tree data structure. The compressed R*-tree data structure is used because of the fact that *realsize(n)* < *nodesize(n)* occurs more often than for R*-tree data structure. We use queries with 1%, 5% and 10% selectivity - meaning random queries which return x% of records from the data structure. We do our experiments with the two different cache sizes - 100 and 500 nodes. All R*-tree data structures used in our experiments are shown in Table 1.

From table 1 we can see creation time of a data structure with common and flexible cache. The creation time of a data structure with the flexible cache takes approximately 3% longer, than the creation of a data structure using the common cache. We can also see that an R*-tree data structure are created generally faster than a compressed R*-tree data structure (using the same data).

Tables 2, 3, and 4 show query processing results for the data structures with 5 millions records. The usage of flexible cache implementation decreases amount of disk accesses and time of query processing in most cases. In some cases the mentioned cache row contention negatively influence the query processing, but generally we can say, that total amount of DAC is reduced.

[3] The experiments were executed on an AMD Opteron(tm) Processor 865 1.80 GHz, 32 GB RAM, 64-bit Operating System Windows Server® Daracenter 2007 with Service Pack 2

TIGER data - Compressed R*-tree						
					creation time [s]	
tree (mil)	height	leaf nodes	inner nodes	size[MB]	flexible cache	common cache
1	3	4442	435	9.52	659.28	625.49
2	3	8891	759	18.8	1321.40	1319.83
5	4	22029	2125	47.1	3507.48	3486.38

TIGER data - R*-tree						
					creation time [s]	
tree (mil)	height	leaf nodes	inner nodes	size[MB]	flexible cache	common cache
1	5	8935	1294	19.9	231.63	225.42
2	5	17914	2015	38.9	482.40	466.15
5	5	44448	5386	97.3	1256.27	1193.23

Table 1. test collections - cache size used - 100 nodes

tree (mil)		R*-tree					
		flexible cache			common cache		
		1%	5%	10%	1%	5%	10%
5	DAC[kB]	58022	216470	370042	58176	211888	370764
	time[s]	3.08	14.08	24.42	3.19	12.50	28.06

Table 2. DAC and time for R*-trees - cache 100

tree (mil)		Compressed R*-tree					
		flexible cache			common cache		
		1%	5%	10%	1%	5%	10%
5	DAC[kB]	30890	112862	195690	33192	113608	198218
	time[s]	23.78	83.39	132.99	33.98	110.05	172.83

Table 3. DAC and time for compressed R*-trees - cache 100

As expected the insert (update or delete) operation is worse than the original solution because of the *cache move*. Inefficiency during *cache move* is from 2% to 5% mainly depending on a number of indexed labels in a data structure. The advantage of the flexible cache is particularly evident in a case of the compressed R*-tree. The flexible cache has aproximately by 30% better the query processing time in our tests in a case of the compressed R*-tree.

5 Future Work

In this article we presented a concept of the flexible cache which reduces number of cache node replacements. It also brings new chalenges such as cache row contention which can occur if some cache row is used more often then the others. We minimize

tree (mil)		Compressed R*-tree (compressed ratio 2)					
		flexible cache			common cache		
		1%	5%	10%	1%	5%	10%
5	DAC[kB]	28982	106486	183726	30054	107016	188298
	time[s]	18.56	85.17	150.22	29.90	119.62	192.92

Table 4. DAC and time for compressed R*-trees - cache 500

the cache row contention using the statistics about the node distribution, however, this approach does not have to work when we have many data structured where one is used more then the others. In the future work we would like to focus on this problem and extend our approach that cache row size is set automatically according to the workload. Therefore, the cache row size will be set according to the cache row usage and not according to the node distribution.

6 Conclusion

In this article we evaluate enhanced cache of the persistent data structure. The proper and right manipulation with the cache is very important for well results from our applications. There is a big performance lost when the cache is inefficient. In the section 3 we descibed our new investigation and solution to have good cache conscious data structure. Then in the serction 4 we confirmed our theroretical thoughts compared to practical tests. We could see that performance was increased and DAC were reduced. Because of some case was no better for new solution, we have open area for new and additional investigation to propose our solution more general.

References

1. T. Johnson and D. Shasha. 2Q: A Low Overhead High Performance Buffer Management Replacement Algorithm. In *Proceedings of the 20th International Conference on Very Large Data Bases*, pages 439–450. Morgan Kaufmann Publishers Inc., 1994.
2. N. Megiddo and D. Modha. ARC: A self-tuning, low overhead replacement cache. In *Proceedings of the 2nd USENIX Conference on File and Storage Technologies*, pages 115–130. USENIX Association, 2003.
3. E. O'neil, P. O'neil, and G. Weikum. The LRU-K page replacement algorithm for database disk buffering. *ACM SIGMOD Record*, 22(2):297–306, 1993.
4. J. Walder, M. Krátký, and R. Bača. Benchmarking Coding Algorithms for the R-tree Compression. In *Proceedings of the 9th Annual International Workshop on Databases, Texts, Specifications and Objects, DATESO*, pages 32–43, 2009.

Modeling of Lexical Relations Between Topics Retrieved from DBLP Journals

Lukáš Hlaváček and Michal Výmola

Department of Computer Science, FEI, VSB - Technical University of Ostrava,
17. listopadu 15, 708 33, Ostrava-Poruba, Czech Republic
{lukas.hlavacek, michal.vymola}@vsb.cz

Abstract. In this paper we present a method for getting topics from aims and scopes in DBLP journals and following construction of their hierarchical order. We focused on semantic relations between topics. Our method is fully automatic but manual cleaning of topic database would lead to much better accuracy. Another purpose of our work is to provide similar system to ACM classification system. We want to provide better, newer and fully automatic system contrary to ACM.

Keywords: DBPL, ACM, Lexical acquisition, Text mining

1 Introduction

This paper describes a method for semantically retrieved topics hierarchy into a graph where the nodes are topics and the edges (also called links) represent relationships between them. We present how to retrieve data from plain text, then algorithm itself an in the last part testing and result analysis. There were not well presented yet methods for automatic topic retrieval from plain text. Of course, our technique is applicable to many other issues requiring word categorization with semantic influence. Existing methods are primarily focused on semantic of ordinary words. In next chapter we mention some of them. But we want to focus in this paper especially on semantic of topics because topics have a special semantic. This paper describes a method for semantically retrieved topics hierarchy into a graph where the nodes are topics and the edges (also called links) represent relationships between them. Levels in our graph represent level in hierarchy.

Our motivation was Widdow's method described in [1], but we apply his technique on topics retrieved from DBLP. For testing we could not use WordNet database as Widdows did because this database contains only information about simple words (synonyms). Topics mostly consists of more words (2-3). That is why we choose ACM classification (latest 1998).

ACM is widely recognized as the premier membership organization for computing professionals, delivering resources that advance computing as a science and a profession, enable professional development and promote policies and research that benefit society.

V. Snášel, J. Pokorný, K. Richta (Eds.): Dateso 2011, pp. 153–160, ISBN 978-80-248-2391-1.

ACM hosts the computing industry's leading Digital Library and Guide to Computing Literature, and serves its global members and the computing profession with journals and magazines, conferences, workshops, electronic forums, and Online Books and Courses.

ACM's first classification system for the computing field was published in 1964. Then, in 1982, the ACM published an entirely new system. New versions based on the 1982 system followed, in 1983, 1987, 1991, and 1998 [8]. ACM classification is already more than 10 years old and does not contain many new topics, for instance *social network* or *wireless networks*.

The DBLP data source, which is representative of conventional database applications, is maintained by a single source. It is one of the best formatted and organized bibliography datasets. DBLP covers approximately 400,000 researchers who have publications in major Computer Science publication venues. Bibliographic datasets have been used for social network analysis, such as studying the structure and the spread of influence in scientific communities.

2 Previous work

In this paper we continue in D. Widdows' work. He presented method for assembling semantic knowledge from a part-of-speech tagged corpus using graph algorithms. His graph model is built by linking pairs of words which participate in particular syntactic relationships.

In this part we present some of most important previous methods. Most work on automatic lexical acquisition has been based at some point on the notion of semantic similarity. Roark and Charniak described a *generic algorithm* for extracting such lists of similar words using the notion of semantic similarity in [7].

Roark and Charniak reported accuracies of 17% and 35%. The early results have been improved upon by Riloff and Jones in [6], where a *mutual bootstrapping* approach is used to extract words in particular semantic categories and expression patterns for recognizing relationships between these words for the purposes of information extraction. The accuracy achieved in this experiment was about 78%.

Another way to obtain word-senses directly from corpora is to use clustering algorithms on feature-vectors. General problem for such clustering techniques lies in the question of how many clusters one should have, i.e. how many senses are appropriate for a particular word in a given domain. Lin's approach to this problem in [10] is to build a *similarity tree* (using what is in effect a hierarchical clustering method) of words related to a target word. Another approach described T. P. Martin and M. Azmi-Murad in [8].

Widdows described a new incremental algorithm for extracting categories of similar words. He defined following method for constructing a hierarchy of words, affecting how they depend on each other.

Definition 1. *Let A be a set of nodes and let $N(A)$ be the neighbours of A. These neigbour nodes are linked to any $a \in A$. (So $N(A) = \bigcup_{a \in A} N(a)$.) The* *best new node is taken to be the node $b \in N(A) \setminus A$ with the highest proportion of links to N (A). More precisely, for each $u \in N(A) \setminus A$, let the affinity between u and A be given by the ratio*

$$af(N(u), N(A)) = \frac{|N(u) \cap N(A)|}{|N(u)|} \tag{1}$$

If $N(u) = \emptyset$ then af(N(u),N(A))=0. The best new node $b \in N(A) \setminus A$ is the node which maximises this affinity score [1].

Here it depends which seed topic is taken as the first one. This algorithm may find for particular topic A some other node B as the best node. But this does not have to find topic A as the best node for topic B if B is a seed topic.

3 Methodology

This chapter describes our approach to retrieve topics from text and how to construct topics hierarchy. Source of our data are manually extracted aims and scopes from DBLP journals. Follows automatic extraction particular topics from these texts.

3.1 The algorithm for creating of dictionary

Let R be the set of abstracts where $r_i \in R$ is abstract of journal. Then let N be the set of improper words (negative dictionary), where $n_m \in T$ is improper word in negative dictionary. Also we define δ, that represents maximal number of words in topic. For example if $\delta = 3$ then topic contantains max three words. Function *split* is standart function of programing language C# and splits text by conjuction.

Algorithm is based on searching of specific important conjunctions in sentences (and, or, comma, semi-comma) and words with a short distance to these conjunctions. These conjunctions have various priority whereas the most important is *and.* Other special characters were removed. In the case of *and* conjunction is necessary to correctly analyze the topic which we do following way:

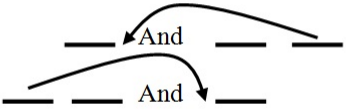


Fig. 1. Dividing of topic containing *and*

In fig.1 is an example of syntactic compound of topic with *and.* Arrows show which word is needed to add and where. For instance *software and hardware architecture* or other example *software architecture and hardware.* In the fig.1 is suggested an algorithm for topics consisting of two words. Similar algorithm is for 3-word topics. It is important to proceed from lexical structure of more-words conjunctions when creating such rules.

It might happen that this method divides the topic into two new topics which were not supposed to be divided, e.g. *Data terminals and printers.* Other conjunctions usually does not have this issue and these conjunctions have for semantic of topics also the same priority. Next step is to save these parsed topics into incident matrix which creates relations between particular nodes. Then we can easily set up level of importance according to frequency of their occurrence in text.

Algorithm 1 The algorithm for create dictionary

```
Require: $\delta = 3$, R=set of abstracts, N=set of improper words
  for all $r_i \in R$ do
    S=split($r_i$, '.') {where S is the set of sentences for $r_i \in R$}
    for all $s_j \in S$ do
      W=split($s_j$,conjunction) {where W is potencial the set of topics for $s_j \in S$}
      for all $w_k \in W$ do
        if $|w_k| <= \delta$ and $w_k \notin N$ then
          Add $w_k$ to T {where T is the set of topics}
        end if
      end for
    end for
  end for
```

3.2 The algorithm for create graph of topics

Let T be the set of topics, where $t_i \in T$ is topic in dictionary of topics. Let R be the set of abstracts, where $r_i \in R$ is abstract of journal. Also we use incidence matrix $I^{m \times n}$, that represents link between words in dictionary.We define ϵ that represents required strength of relationship between two words in incidence matrix.

In the first phase we analyze each journal separately and we search for one, two or three-word topics.

4 Testing

We identified about 750 journals in DBLP.For testing we used 500 aims and scopes from these journals and containing about 90 000 words. This text was parsed by algorithm 1 into about 4000 topics. During testing we found out that

Algorithm 2 The algorithm for searching of similar topics

```
Require: $I^{m \times n}$=empty matrix, $m = n = |T|$, $\epsilon$=required strength of relationship,
  A=the set of seed topics, x=number of topics
  for all $r_i \in R$ and do
    S=split($r_i$, '.') {where S is the set of sentences for $r_i \in R$}
    for all $s_j \in S$ do
      W=split($s_j$,conjunction) {where W is potencial the set of topics for $s_j \in S$}
      Compute $Z = W \cap T$ {where Z is the set of topics in sentence}
      for all $z_n, z_m \in Z$ do
        Create link between $z_m, z_n$ in I, where $i_{z_m z_n} = i_{z_m z_n} + 1$.
      end for
    end for
  end for
  for all $i_{mn} \in I$ do
    if $i_{mn} < \epsilon$ then
      $i_{mn} = 0$
    end if
  end for
  while $|A| < x$ do
    Compute $N(A)$ from matrix I
    for all $b_i \in N(A) \setminus A$ do
      Compute $N(b_i)$ from matrix I
      Compute $af(N(b_i), N(A))$
    end for
    Add $b_i$ with the highest affinity to A.($b_i \cup A$)
  end while
```

for our topic-database size we are obtaining the best results if we take first 1000 topics with the highest frequency of occurrence. So there is about 3000 topics left but their rating is too low to affect the result significantly.

We achieved this result by following way: first we took 400 topics, but in this setting we found only few related topics. Then why we doubled size of topics. In this way we continued up to 2000 topics and we were observing generated groups. We were also focusing on reliability of topics in particular groups so we do not obtain topics out of seed topic in group. We set a border up to 10 good topics in each group, if possible. However for some of topics algorithm finished in lower number. From all results we found that for our database is optimal to take first 1000 topics with the highest number of occurrences. We compared our results to ACM hierarchy.

5 Conlusion

Most from previous works were using WordNet database for testing. Because topics are semantically different we could not use this one but instead we used ACM hierarchy database which is a database of topics hierarchically ordered. It is possible to download also in XML so it is very easy to work with it. Even we

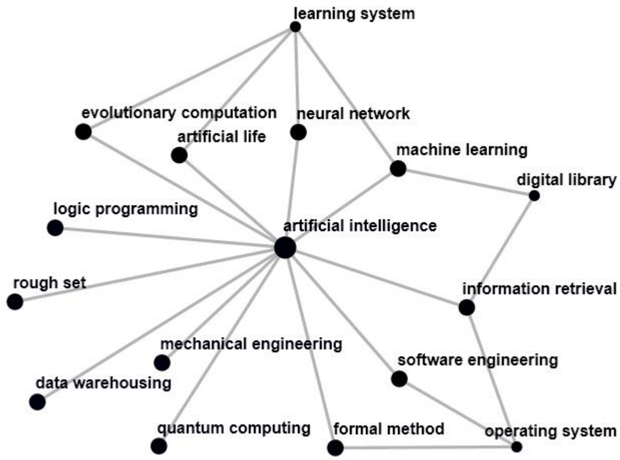


Fig. 2. Graph generated for topic *Artificial Intelligence*

are not able to measure accuracy as a single number, we can pursue conformity to ACM.

Result is strongly affected by the age of last release of ACM hierarchy (in 1998) when many topics did not exist, for instance social networks or business intelligence. Our algorithm found about 5000 automatically generated topics from 500 journals. There were manually found approximately 10% of topics which were not good topics. Latest ACM database consists of 1200 topics. We choose some topics common for ACM and our results and compared them. Since we found only about 20 percent of topics same in both databases it does not make much sence to compare them together. The difference between both databases is that for instance in ACM there is a single topic *graphic* but our algorithm generates *computer graphic* and 8 other topics containing word *graphic*. This is also a reason why is our hierarchy so much larger.

Figure 4 shows how algorithm is adding every new topic into hierarchy. First are added topics as first level which were directly connected to the seed topic, then to the level 2 were added topics with the highest ratio to topics from first level. This algorithm iterates the same for next levels.

In Table 1 we present results for few randomly chosen seed topics. We can see that our method generates corectly related topics to the seed topic. However there might be some topics not related to the seed word and these topics are marked in table as italic font. Compare to ACM we obtained much better related topics in groups. More aims and scopes we have better results we get.

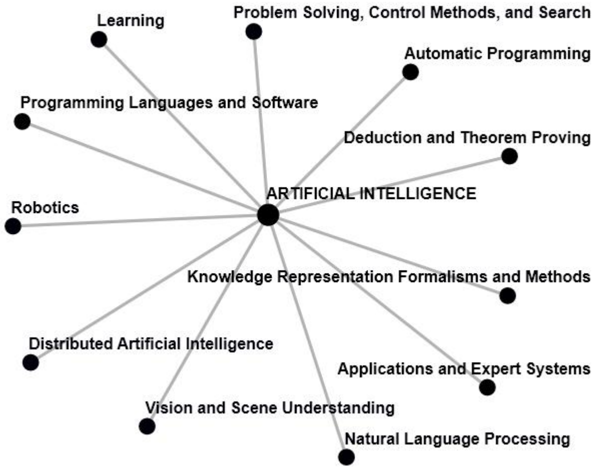

Learning
Problem Solving, Control Methods, and Search
Automatic Programming
Programming Languages and Software
Deduction and Theorem Proving
ARTIFICIAL INTELLIGENCE
Robotics
Knowledge Representation Formalisms and Methods
Distributed Artificial Intelligence
Applications and Expert Systems
Vision and Scene Understanding
Natural Language Processing

Fig. 3. ACM graph for topic *Artificial Intelligence*

References

1. Widdows D., Beate Dorow: A Graph Model for Unsupervised Lexical Acquisition. *COLING, 2002.*
2. Widdows D. and Ferraro K.: Semantic vectors: A scalable open source package and online technology management application. *In Proceedings of the sixth international conference on Language Resources and Evaluation,2008.*
3. Widdows, D.: Orthogonal negation in vector spaces for modeling word-meanings and document retrieval. *In Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics,2003.*
4. Rieffel, E.: Certainty and uncertainly in quantum information processing. *In Bruza et al. (2007)*, 134-141.
5. Widdows, D. and Bruza P.: Quantum information dynamics and open world science. *In Bruza et al. (2007)*,126-133.
6. Ellen Riloff and Rosie Jones: Learning dictionaries for information extraction by multi-level bootstrapping. *In Proceedings of the Sixteenth National Conference on Artificial Intelligence, 1999*
7. Brian Roark and Eugene Charniak: Noun-phrase co-occurence statistics for semiautomatic semantic lexicon construction. *In COLING-ACL,1998*
8. Trevor P. Martin, Masrah Azmi-Murad: An Incremental Algorithm to find Asymmetric Word Similarities for Fuzzy Text Mining. *WSTST 2005*, 838-847
9. Communications of the ACM, New York, NY, USA
10. Dekang Lin: Automatic retrieval and clustering similar words,*In COLINGACL,Montreal,Athens*

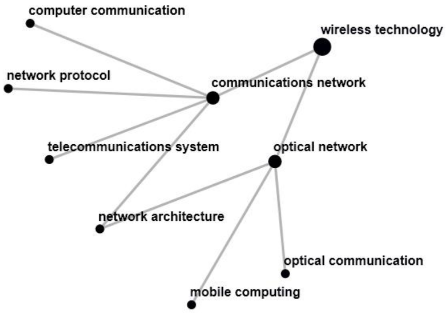


Fig. 4. Hierarchical graph generated for *Wireless technology*

Table 1. Groups of topics for chosen seed topics

Seed topic	Topics
artificial intelligence	formation retrieval, *digital library*, *mechanical engineering*, *data warehousing*, logic programming, machine learning, evolutionary computation, quantum computing, artificial life, learning system, neural network
wireless	fspread-spectrum system, cellular system, *ip*, adaptive antenna, network protocol, telecommunications network, computer communication, *lan*, *man*, satellite network, wireless computing
computer graphic	image processing, computer vision, *speech recognition*, data mining, machine learning, digital library, evolutionary computation, artificial life, learning system, data warehousing
mathematical method	computational method, numerical mathematic, computational mathematic, stochastic process, set theory, probability theory, matroid theory, computer scientist, coding theory, differential equation,
process engineering	production engineering, electronics engineering, unmanned system, electrical engineering, chemical engineering, mechanical engineering, traffic engineering, *applied mathematic*
language research	language technology, human-computer interaction, decision support, computational linguistic, spatial reasoning, qualitative reasoning, *global warming*, semantic web, object-oriented programming
combinatorial optimization	randomized algorithm, mathematical programming, graph algorithm, indexing information, data analysis, *computer vision*, information fusion, *database design*, *web mining*, risk analysis

Overview and Comparison of Plagiarism Detection Tools

Asim M. El Tahir Ali, Hussam M. Dahwa Abdulla, and Václav Snášel

Department of Computer Science, VŠB-Technical University of Ostrava,
17. listopadu 15, Ostrava - Poruba, Czech Republic
asim070@yahoo.com, hussamdahwa@hotmail.com, vaclav.snasel@vsb.cz

Abstract. In this paper we have done an overview of effective plagiarism detection methods that have been used for natural language text plagiarism detection, external plagiarism detection, clustering-base plagiarism detection and some methods used in code source plagiarism detection, also we have done a comparison between five of software used for textual plagiarism detection: (PlagAware, PlagScan, Check for Plagiarism, iThenticate and PlagiarismDetection.org), software are compared with respect of their features and performance.

1 Introduction

"Plagiarism, the act of taking the writings of another person and passing them off as one's own. The fraudulence is closely related to forgery and piracy-practices generally in violation of copyright laws." Encyclopedia Britannica [5].

Plagiarism can be considered as one of the electronic crimes, like (computer hacking, computer viruses, spamming, phishing, copyrights violation and others crimes). Plagiarism defined as the act of taking or attempting to take or to use (whole or parts) of another person's works, without referencing or citation him as the owner of this work. It may include direct copy and paste, modification or changing some words of the original information from the internet books, magazine, newspaper, research, journal, personal information or ideas.
According to the Merriam-Webster Online Dictionary, to "plagiarize" means:

- To steal and pass off (the ideas or words of another) as one's own.
- To use (another's production) without crediting the source.
- To commit literary theft.
- To present as new and original an idea or product derived from an existing source.

Also according to Turnitin.com, plagiarism.org and Research Resources this are considered plagiarism:

- Turning in someone else's work as your own.
- Copying words or ideas from someone else without giving credit.
- Failing to put a quotation in quotation marks.

V. Snášel, J. Pokorný, K. Richta (Eds.): Dateso 2011, pp. 161–172, ISBN 978-80-248-2391-1.

- Giving incorrect information about the source of a quotation.
- Changing words but copying the sentence structure of a source without giving credit.
- Copying so many words or ideas from a source that it makes up the majority of your work, whether you give credit or not (see our section on "fair use" rules).

Plagiarism can be classified into five categories:

1. Copy & Paste Plagiarism.
2. Word Switch Plagiarism.
3. Style Plagiarism.
4. Metaphor Plagiarism.
5. Idea Plagiarism.

There are two types of plagiarism are more occurs:

1. Textual plagiarisms: this type of plagiarism usually done by students or researchers in academic enterprises, where documents are identical or typical to the original documents, reports, essays scientific papers and art design.
2. A source code plagiarism: also done by students in universities, where the students trying or copying the whole or the parts of source code written by someone else as one's own, this types of plagiarism it is difficult to detect.

2 Why Plagiarism Detection is Important

In some of the academic enterprises like universities, schools and institutions, plagiarism detection and prevention became one of the educational challenges, because most of the students or researchers are cheating when they do the assigned tasks and projects. This is because a lot of resources can be found on the internet. It is so easy to them to use one of the search engines to search for any topic and to cheat from it without citing the owner of the document. So it is better and must all academic fields they should have to use plagiarism detection soft-wares to stop or to eliminate students cheating, copying and modifying documents when they know that they will be found.

Some types of plagiarism acts can be detected easily by using some of the recent plagiarism detection soft-wares available on the market or over the internet. However for some of the expert plagiarism who is using some of the anti-plagiarism soft-wares which are available over the internet, it needs more efforts to detect the plagiarism or cannot be detected at all.

Plagiarism is practiced not only by student but also there are some staff members who like to publish papers in which some parts are directly copied or partially modified to be one of the famous people.

There is a big number of plagiarism soft-wares used for plagiarism detection and many of detection tools have been developed by researchers but still they have some limitations as they cannot prove or they show evidence that the documents has been plagiarized from another document or sources it only shows the similarity and give hints to some other documents. This is if the paper has been published globally in some international journal, but some of universities and some of the research centers still do not taking any action against plagiarism detection which help people to cheat more and more.

So still now by using the recent detection software, plagiarism can not 100% be detected?

Copyrights and legal aspects for use of published documents also can be covered by using plagiarism software, so it can show whether this person has legally or illegally copied the documents or not and it also show the whether this person has permission from the owner to use this document or not.

Plagiarism detection is also one of the most important issues to journals, research center and conferences; they are using advanced plagiarism detection tools to ensure that all the documents have not been plagiarized, and to save the copyrights from violation for the publishers.

3 Plagiarism Detection Methods

In both the textual document plagiarism and source code plagiarism, detection can be either: Manual detection or automatic detection.

- Manual detection: done manually by human, its suitable for lectures and teachers in checking student's assignments but it is not effective and impractical for a large number of documents and not economical also need highly effort and wasting time.
- Automatic detection (Computer assisted detection): there are many software and tools used in automatic plagiarism detection, like PlagAware, PlagScan, Check for Plagiarism, iThenticate, PlagiarismDetection.org, Academic Plagiarism, The Plagiarism Checker, Urkund, Docoloc and more.

3.1 Textual Plagiarism

Many of researchers are developed a set of tools used in textual automatic detection like:

Grammar-based method The grammar-based method is important tool to detect plagiarism. It focuses on the grammatical structure of documents, and this method uses a string-based matching approach to detect and to measure similarity between the documents. The grammar- based methods is suitable for detecting exact copy without any modification, but it is not suitable for detecting modified copied text by rewriting or switching some words that has the same meaning. This is considered as one of this method limitations [4].

Semantics-based method The semantics-based method, also considered as one of the important method for plagiarism detection, focuses on detecting the similarities between documents by using the vector space model. It also can calculate and count the redundancy of the word in the document, and then they use the fingerprints for each document for matching it with fingerprints in other documents and find out the similarity. The semantic-based method is suitable for non partial plagiarism as mentioned before use the whole document and use vector space to match between the documents, but if the document has been partially plagiarized it cannot achieve good results, and this is considered as one of the limitations of this method, because it is difficult to fix the place of copied text in the original document [4, 1].

Grammar semantics hybrid method Grammar semantic hybrid method is considered as the most important method in plagiarism detecting for the natural languages. This method, so effective in achieving better and improving plagiarism detection result, is suitable for the copied text including modified text by rewriting or switching some words that have the same meaning, which cannot be detected by grammar-based method. It also solves the limitation of semantic-based method. Grammar semantic hybrid method can detect and determine the location of plagiarized parts of the document, which cannot be detected by semantic-based method, and calculating the similarity between documents [4, 1].

External plagiarism detection method The external plagiarism detection relies on a reference corpus composed of documents from which passages might have been plagiarized A passage could be made up of paragraphs, a fixed size block of words, a block of sentences and so on. A suspicious document is checked for plagiarism by searching for passages that are duplicates or near duplicates of passages in documents within the reference corpus. An external plagiarism system then reports these findings to a human controller who decides whether the detected passages are plagiarized or not. A naive solution to this problem is to compare each passage in a suspicious document to every passage of each document in the reference corpus. This is obviously prohibitive. The reference corpus has to be large in order to find as many plagiarized passages as possible [20].

This fact directly translates to very high runtimes when using the naive approach. External plagiarism detection is similar to textual information retrieval (IR) [3]. Given a set of query terms an IR system returns a ranked set of documents from a corpus that best matches the query terms. The most common structure for answering such queries is an inverted index. An external plagiarism detection system using an inverted index indexes passage of the reference corpus' documents.

Such a system was presented in [7] for finding duplicate or near duplicate documents.

Another method for finding duplicates and near duplicates is based on hashing or fingerprinting. Such methods produce one or more fingerprints that describe the content of a document or passage. A suspicious document's passages are compared to the reference corpus based on their hashes or fingerprints. Duplicate and near duplicate passages are assumed to have similar fingerprints. One of the first systems for plagiarism detection using this schema was presented in [2]. External plagiarism detection can also be viewed as nearest neighbor problem in a vector space R^d.

Clustering in plagiarism detection Document clustering is one of the important techniques used by information retrieval in many purposes; it has been used in summarization of the documents to improve the retrieval of data by reducing the searching time in locating the document. It is also used for result presentation. Document clustering is used in plagiarism detection to reduce the searching time. But still now in clustering there are some limitations and difficulties with time and space [8].

Most of the above methods have been used by textual documents plagiarism detection.

3.2 Source code plagiarism

Source code plagiarism or it called programming plagiarisms usually done by students in universities and schools can be defined act or trial to use, reuse, convert and modify or copy the whole or the part of the source code written by someone else and used in your programming without citation to the owners. Source code detection mainly requires human intervention if they use Manual or automatic source code plagiarism detection to decide or to determine whether the similarity due to the plagiarism or not. Manual detection of source code in a big number of student homework's or project it is so difficult and needs highly effort and stronger memory, it seems that impossible for a big number of sources.

Plagiarism detection system or algorithms used in source-code similarity detection can be classifies according to Roy and Cordy [9] can be classified as based on either:

- 'Strings - look for exact textual matches of segments, for instance five-word runs. Fast, but can be confused by renaming identifiers'.
- "Tokens - as with strings, but using a lexer to convert the program into tokens first. This discards whitespace, comments, and identifier names, making the system more robust to simple text replacements. Most academic plagiarism detection systems work at this level, using different algorithms to measure the similarity between token sequences".
- "Parse Trees - build and compare parse trees. This allows higher-level similarities to be detected. For instance, tree comparison can normalize conditional statements, and detect equivalent constructs as similar to each other".
- "Program Dependency Graphs (PDGs) - a PDG captures the actual flow of control in a program, and allows much higher-level equivalences to be located, at a greater expense in complexity and calculation time".

- "Metrics - metrics capture 'scores' of code segments according to certain criteria; for instance, "the number of loops and conditionals", or "the number of different variables used". Metrics are simple to calculate and can be compared quickly, but can also lead to false positives: two fragments with the same scores on a set of metrics may do entirely different things ".
- "Hybrid approaches - for instance, parse trees + suffix trees can combine the detection capability of parse trees with the speed afforded by suffix trees, a type of string-matching data structure".

There are many methods developed by researcher for source code plagiarism detection like:

- Cynthia Kustanto and Inggriani Liem: they developed a tool for automatic source code detection call Deimos, used in source plagiarism detection, to provide a clear readable form and to erase the displayed result. It was develop to be used with LISP and Pascal programming languages. The time consumed by this tool for section a number of 100 LISP was efficient [11].
- Boris Lesner, Romain Brixtel, Cyril Bazin and Guillaume Bagan: they introduced a new frame work named A Novel Framework to Detect Source Code Plagiarism, mainly used in detection of four type of code source plagiarisms which are change the code name, rebuilt or recoded again, move, add, change and remove the code and replace some text from place to place with the code. A bottom-up approach has been implemented to six steps which are: 1- first the Pre-flattering the source code: they use common method in filtering a source code that by indicating and rename each alphanumerical string in the code. 2- Second they segment the source code to segmentation and measure the similarity on it 3- thirdly they matched each segment and reposted it for filtering. 4-5: Fourthly the use matrix M that have been used in filtering in evaluation of the document 6- In this stage start to analysis the original document according to the evaluation done by document wise distance. This method has been applied to copra languages and shows a great result [12].
- Ameera Jadalla and Ashraf El Nagar: They develop Plagiarism Detection Engine used for detection of source code plagiarism for Java (PDE4Java). The proposed search engine divided in to three steps 1- step one is the process of the tokenization for the Java code 2- second step is to find and measure the similarity between the original code and the tokenized code 3- lastly is to cluster the Java code in order to be used in plagiarism detection as reference. This search engine can be used with all programming language due to its flexibility. Report can show for each cluster code besides the textual [10].

4 Comparisons

We compare the plagiarism software used in textual and source code plagiarism into two categories: first according to features and secondly according to performance [6]. Qualitative comparison used in comparing the features of software, where we are looking for properties of the tools. Quantitative comparisons used in comparing the performance of software, where it depends on the result. Here comparison of some textual soft-wares:

4.1 PlagAware

Is an online-service used for textual plagiarism detection, which allows and offers some services for the user for example can search, find, analyze and trace plagiarism in the specified topic similar to the topics, PlagAware is a search engine, which is considered as the main element, which is strong in detecting typical contents of given texts. It uses the classical search engine for detecting and scanning plagiarism, and provide different types of report that help the user or the document owner to decide that is his document has been plagiarized or not. The two primary fields for PlagAware plagiarism search engine is webpage monitoring for theft contents and transmitted text assessment. In [13], there are three application fields of PlagAware [14]:

- Tracing content theft: Webmaster can use PlagAware for detecting and tracing plagiarisms of websites, in order to find out the plagiarized or the copied contents. PlagAware is considered as strong total solution software systems, which allow the operators of websites to do an automatic observation of their own pages against possible content theft.
- PlagAware is used in search for plagiarisms of student's academic documents and analyze them. Also it is used to assess plagiarism, also to follow and prove the origin of the works including all of academic documents. It generates report that helps them to fast detection of plagiarism.
- Proof of authorship is also provided by PlagAware: it became more important to the authors to ensure that the authorization have been granted to their publication including all types of publication this gives them additional competitive advantage and increase the value of your work.

The main features of PlagAware are [15]:

- Database Checking: PlagAware is a search engine that allows the user to submit his document and Plagaware start searching over the internet. So mainly it does not have local database but it offers checking other database that are available over the internet.
- Internet Checking: PlagAware is an online application and it considered as one of search engine, allows the student or webmaster to upload and check their academic documents, homework, manuscript and articles to be searched against plagiarism over world wide web.ans also provides a webmaster to have capability to do automatic observation of their own page against possible contents theft.
- Publications Checking: PlagAware: support mainly used in academic filed so it provides checking of most types of submitted publication like homework, manuscript, documents, including, books, articles, magazines, journals, editorial and PDFs etc.

- Synonym and Sentence Structure Checking: PlagAware does not support synonym and sentence structure checking.
- Multiple Document Comparison: PlagAware offers comparison of multiple documents.
- Supported Languages: PlagAware supports German as primary language, English and Japanese as secondary languages.

4.2 PlagScan

PlagScan is online software used for textual plagiarism checker. PlagScan is often used by school and provides different types of account with different features. PlagScan use complex algorithms for checking and analyzing uploaded document for plagiarism detection, based on up-to-date linguistic research. Unique signature extracted from the document's structure that is then compared with PlagScan database and millions of online documents. So PlagScan is able to detect most of plagiarism types either directs copy and paste or words switching, which provides an accurate measurement of the level of plagiarized content in any given documents [16]. The Main features of PlagScan are [15]:

- Database Checking: PlagScan it has own database that include millions documents like (paper, articles and assignments), and articles over World Wide Web. So it offers database checking whether locally or others database over the internet.
- Internet checking: PlagScan is an online checker so it provides internet checking to all submitted documents. Whether that the document available on the internet or available in the local database or cached.
- Publications Checking: PlagScan: is mainly used in academic filed so it provides checking most types of submitted publication like documents, including, books, articles, magazines, journals, newspapers, PDFs etc. online only.
- Synonym and Sentence Structure Checking: PlagScan does not support synonym and sentence structure checking but provides Integration via application programming interface in your existing content management system or learning management system possible.
- Multiple Document Comparison: CheckForPlagiarism.net offers comparison of multiple documents in parallel.
- Supported Languages: PlagScan supports all the language that use the international UTF-8 encoding and all language with Latin or Arabic characters can be checked for plagiarism.

4.3 CheckForPlagiarism.net

CheckForPlagiarism.net was developed by a team of professional academic people and became one of the best online plagiarism checkers that used to stop or prevention of online plagiarism and minimizes its effects on academic integrity. In order to maximize the accuracy CheckForPlagiarism.net has used the some methods like document fingerprint and document source analysis to protect document against plagiarism. The fingerprint-based approach used to analyze and summarize collection of document and create a kind of fingerprint for it. Some of numerical attributes can be used by fingerprint that somehow reflects in the structure of the document. So by creating fingerprint for each document with some of numerical attributes for each document in the collection, we can easily find the matching or the similarity between documents across billions of articles. Using this feature by CheckForPlagiarism.net increased the efficiency in detecting most types of plagiarisms [17]. The main features of CheckForPlagiarism.net are [15]:

– Database Checking: CheckForPlagiarism.ne uses its own database that include millions documents like (paper, articles and assignments), and articles over World Wide Web. So it offers fast and reliable depth database checking, also provides checking through all other databases in different fields like medical database, law- related database and other specialty and generalized databases.
– Internet Checking: CheckForPlagiarism.ne: live(online) and cached links to websites used for extensive internet checking to all submitted documents. One more advantage is that it can still check your documents against if a website that is no longer online, this include all contents of website like forums, message boards, bulletin boards, blogs, and PDFs etc., all this check is done automatically and in (almost) real-time.
– Publications Checking: CheckForPlagiarism.net offers detailed and deep checking of most types of submitted publication documents, including, books, articles, magazines, journals, newspapers, PDFs etc. this is done whether the publications is available online (active on the internet) or not available on the internet offline (store paper based).
– Synonym & Sentence Structure Checking: CheckForPlagiarism.net is said to have a sole advantage, that other soft-wares do not support, which is the fact that it uses a "patented" plagiarism checking approach. In which the sentence structure of a document is checked to ensure improper paragraphing and thus is susceptible to plagiarism. Also a synonym check is done to words and phrases to identify any attempt of plagiarism.
– Multiple Document Comparison: CheckForPlagiarism.net can compare a set of different documents simultaneously with other documents and can diagnose different type of plagiarisms at the sometimes [15].
– Supported Languages: CheckForPlagiarism.net supports English languages, Spanish, German, Portuguese, French, Italian, Arabic, Korean, and Chinese languages [15].

4.4 iThenticate

iThenticate one of the application or services designed especially for the researchers, authors' publisher and other. It provided by iParadigms that have introduced Turnitin in 1996 to become the online plagiarism detection. It is designed to be used by institutions rather than personal, but lastly they provided a limit service for single plagiarism detection user like master and doctoral students and this allows them to check a single document of up to 25,000 words. So they can use this service to insure or to check their draft thesis whether containing correct citation and content originality [18]. The main features of iThenticate are [15]:

– Database Checking: iThenticate used its own database that contain millions of documents like (books, paper, essays, articles and assignments), with a large number of this documents that have been stored in iThenticate database locally, allowing the users who have account to do either online and offline comparison of submitted documents against it and to identify plagiarized content.
– Internet Checking : iThenticate, is considered as the first online plagiarism checker that provides live and cached links to websites and database to have extensive internet checking to all submitted documents. This Provides deep internet checking. One more advantage is that it can still check your documents even if a website is no longer online, this include all contents of website like forums, message boards, bulletin boards, blogs, and PDFs etc., all this check is done automatically and in (almost) real-time.
– Publications Checking: iThenticate offers an online and offline detailed and depth checking most types of publication like documents, including, books, articles, magazines, journals, newspapers, website and PDFs etc.
– Synonym & Sentence Structure Checking: Not supported by iThenticate.
– Multiple Document Comparison: iThenticate offers two types of document comparison document to document and multiple documents checking against database and also direct source comparison word to word also.
– Supported Languages: iThenticate supports more than 30 languages, it mean that it supports most of languages likes ”English, Arabic, Chinese, Japanese, Thai, Korean, Catalan, Croatian, Czech, Danish, Dutch, Finnish, French, German, Hungarian, Italian, Norwegian, Polish, Portuguese, Romanian, Serbian, Slovak, Slovenian, Spanish, Swedish, Greek, Hebrew, Farsi, Russian, and Turkish.” [18].

4.5 PlagiarismDetection.org

PlagiarismDetection.org: an online service provides high level of accuracy result in plagiarism detection. Mainly designed to help the teachers and student to maintain and to ensure or prevent and detect plagiarism against their academic documents. It provides quickly detect plagiarism with high level of accuracy [19]. The main features of PlagiarismDetection.org [15]:

– Database Checking: PlagiarismDetection.org used it own database that contains millions of documents like (books, paper, essays, articles and assignments).
– Internet Checking: PlagiarismDetection.org is an online plagiarism detector, so it is mainly based on the internet checking and is faster in plagiarism detection, it does not support offline detection.

- Publications Checking: PlagiarismDetection.org offers the students and teachers to check their publication against the published document and support most types of publication.
- Synonym & Sentence Structure Checking: PlagiarismDetection.org not supports Synonym & Sentence Structure Checking.
- mMultiple Document Comparison: PlagiarismDetection.org does not support multiple document comparison but it takes long time to return the result.
- Supported Languages: PlagiarismDetection.org supports English languages and all languages that using Latin characters.

Table1: Summarization of the comparison according to the software features: Key of the table figure1: The following expressions denote that: ***** Excellent, **** Very good, *** Good, ** Acceptable and * Poor.

Table 1. Comparsion of the software

Features	PlagAware	PlagScan	iThenticate	CheckFor Plagiarism.net	Plagiarism detecting.org
Database Checking (online and offline)	*****	*****	*****	****	****
Internet Checking	*****	*****	*****	*****	*****
Publication Checking	*****	*****	*****	***	***
Multiple document comparison	*****	*****	*****	****	****
Multiple languages support	*****	*****	*****	*****	*****
Sentence structure and synonym checking	****	**	****	*****	**

5 Conclusions

The comparison of the software shown that still now their no software that can detect or to prove that the document has been plagiarize 100%, because each software and tool has advantages and limitation, according to the following features and performance, (Database checking whether locally or online, internet checking whether is online or offline, publications documents checking and supported types, capabilities of multiple document comparison, supported languages and synonym and sentence structure checking) we ranked them as follows, starting from the best, PlagAware, iThenticate, PlagScan, CheckForPlagiarism.net and lastly PlagiarismDetection.org. Academic enterprises can use one of the above software in their detecting of plagiarism but, due the limitation of most of the software and importance of plagiarism detection to the academic fields we suggest some rules that can be used to limit or to reduce student plagiarism teacher should educated student about plagiarism and its impact, copy right, citation and ownership.

In future work we may want to extend our comparison to larger and more varied set of real-life data and to extent our comparison to include more textual plagiarism software like Academic Plagiarism, The Plagiarism Checker, Urkund and Docoloc, also to extended to include code source plagiarism detection software according to Supported languages, Extendibility, Presentation of results, Usability, Exclusion of template code, Exclusion of small files, Historical comparisons, Submission or file- based rating, Local or web-based and Open source

References

1. Alzahrani, S. Salim, N. Abraham, A. Understanding Plagiarism Linguistic Patterns, Textual Features and Detection Methods, preprint 2011.
2. Brin, S. and Davis, J. and Garcia-Molina, H. Copy Detection Mechanisms for Digital Documents. In: ACM International Conference on Management of Data (SIGMOD 1995), May 22-25, 1995
3. Baeza-Yates, R. & Ribeiro-Neto, B. Modern Information Retrieval. Addison-Wesley, 1999.
4. Bao Jun-Peng, Shen Jun-Yi, Liu Xiao-Dong, Song Qin-Bao, A Survey on Natural Language Text Copy Detection, Journal of Software, 2003, vol.14, No.10, pp. 1753-1760.
5. Encyclopedia Britannica, http://www.britannica.com/EBchecked/topic/462640/plagia (last access February 7, 2011)
6. Hage, J. Rademaker, P. Vugt, N. A comparison of plagiarism detection tools, Technical Report UU-CS-2010-015,June 2010, Department of Information and Computing Sciences Utrecht University, Utrecht, The Netherlands.
7. Hoad, T. C., Zobel, J. Methods for Identifying Versioned and Plagiarized Documents. JASIST 54(3): 203-215 (2003)
8. Jain, K. Murty, M. N., Flynn, P. J. Data clustering: a review. ACM Computing Surveys, 31(3):264–323, 1999.
9. http://www.cs.queensu.ca/queensu.ca/TechReports/Reports/2007-541.pdf. (last access February 7, 2011)
10. Jadalla, A. Elnagar, A. PDE4Java: Plagiarism Detection Engine for Java source code: a clustering approach. IJBIDM 3(2):121-135 (2008)
11. Kustanto, C. Liem, I. Automatic Source Code Plagiarism Detection. SNPD 2009:481-486.
12. Lesner, B. Brixtel, R. Bazin, C. Bagan, G. A novel framework to detect source code plagiarism: now, students have to work for real! SAC 2010:57-58.
13. http://www.plagaware.com/about_plagaware (last access February 7, 2011)
14. http://www.plagaware.com/about_plagaware/application (last access February 7, 2011)
15. http://plagiarism-checker-review.toptenreviews.com/index.html (last access February 7, 2011)
16. http://www.plagscan.com. (last access February 7, 2011)
17. http://www.checkforplagiarism.net (last access February 7, 2011)
18. http://www.ithenticate.com/index.html (last access February 7, 2011)
19. http://www.plagiarismdetection.org (last access February 7, 2011)
20. Zechner, M., Muhr, M., Kern, R., Michael, G. External and intrinsic plagiarism detection using vector space models. In: Proceedings of the SEPLN'09 Workshop on Uncovering Plagiarism, Authorship and Social Software Misuse. pp. 4755 (2009).

Précis Index Implementation for Efficient Fulltext Data Mining

Michal Kopecký[1] and Martin Čermák

Department of Software Engineering,
[1]Faculty of Mathematics and Physics, Charles University Prague
Malostranské nám. 25, 118 00 Prague, Czech Republic
michal.kopecky@mff.cuni.cz, cermak23@gmail.com

Abstract. Précis system has been designed for text based searching over relational database. Unlike the common approach, this system allows user to search requested data over a whole database, not only within one table. System takes queries formulated in free-form and produces rows containing information corresponding to the query and also information associated to them within the database. This paper presents implementation of the index, suitable for efficient searching in the Oracle relational database management system. Implementation provides SQL query language extension for searching desired data.

Keywords: Information Retrieval, Précis index, Relational Databases, Data Mining.

1 Introduction

In the relational databases application designers have to define appropriate set of indexes to speed-up the searching process and data manipulation. These indexes are usually implemented using redundant B^+ trees. It is possible to find also different index type implementations recently. Some of them try to overcome B+ tree limitations as requirement for high selectivity while other try to support searching completely different types of data (XML, images, texts, geographical data etc.).

Standard searching within the database requires that application developers understand the database structure. The Précis system invented in [1] allows users to query the database without exact knowledge about the database and provides them with exhaustive answer. Users formulate queries using keywords and the system looks for all relevant data within the database independently on the fact in which table and/or column the information are stored. Moreover the system is able to trace foreign key – primary key links within the database and find out also related data. The result can be then presented as a hierarchy of resulting rows, or can be formatted to sentences in natural language.

However, the existing built-in indexes doesn't support search over all columns of all tables and naïve implementation of the system using LIKE operator will be extremely

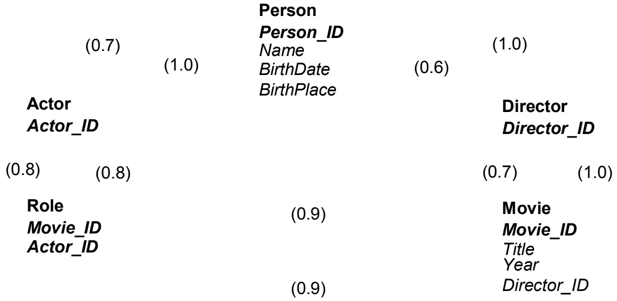
V. Snášel, J. Pokorný, K. Richta (Eds.): Dateso 2011, pp. 173–180, ISBN 978-80-248-2391-1.

slow and ineffective. This paper presents the implementation of the Précis index in the Oracle database system using its extensibility features.

Following chapter describes the Précis indexing in more details. The paper concerns on brief implementation description and obtained performance results. More details about the implementation can be found in [2]

2 Précis system

The primary data are stored in classical database and can be processed by standard database application. The Précis querying uses data stored in all tables and columns and takes into account associations between tables derived from foreign key references present in the database. References, tables and columns in the database can be additionally rated by weights, represented by numbers $\boldsymbol{w} \in <0;1>$. These weights can be either global or associated to individual users. The Précis query $\boldsymbol{q}$ consists of a set of keywords $\boldsymbol{q}=\{\boldsymbol{k}_1, \boldsymbol{k}_2, \ldots, \boldsymbol{k}_n\}$. According to the given query the system finds out corresponding records and related information. Let suppose following database structure:



Picture 1. Sample database structure

The broader arrows represent foreign keys in the database, narrower dotted arrows represent opposite associations. Numbers in square brackets represent ratings for associations. Having such a database, the query $\boldsymbol{q}$={“*Woody*”, “*Allen*”} could provide the result (without text formatting) in form:

Director: Woody Allen | December 1st 1935 | Brooklyn | New York | USA
Movie: Match Point | 2005
Movie: Melinda and Melinda | 2004
Movie: Anything Else | 2003
Role: Hollywood Ending | 2002
Role: The Curse of the Jade Scorpion | 2001

Authors of the Précis system have proposed the query evaluation is in a sequence of four successive steps: ***First*** – create list of all occurrences of all query terms in the database D. This step provides subschema D’ of all tables containing those terms. ***Second*** – determine the subset of the database schema accessible from D’ using associations. This step provides subschema D”. ***Third*** – fill database **D**” with the schema D” with retrieved data. ***Last*** – optionally transform data in **D**” to the natural language using text templates.

Each record in the result database obtains a rating according to its relation to data found in the first step. The rating of the row accessible through a chain of association is computed as a product of ratings assigned to corresponding tables, columns and association. The result then contains all data with rating exceeding given threshold.

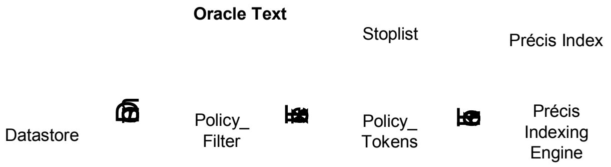
3 Implementation

Our concern was on efficient inverted index building in one of most used RDBMS in the enterprise segment – the Oracle database – and on query evaluation using this index. The Oracle database was chosen because it provides developers with necessary background for creating user-defined index types.

The basic requirements on the Précis index were as follows:

- Each index should contain a set of columns stored in arbitrary number of tables within the database.
- Each user should be able to create any number of indexes.
- One column can be assigned to any number of defined indexes.
- The user interface for index manipulation should be as user-friendly as possible Querying the database should be available through the Oracle SQL query language.

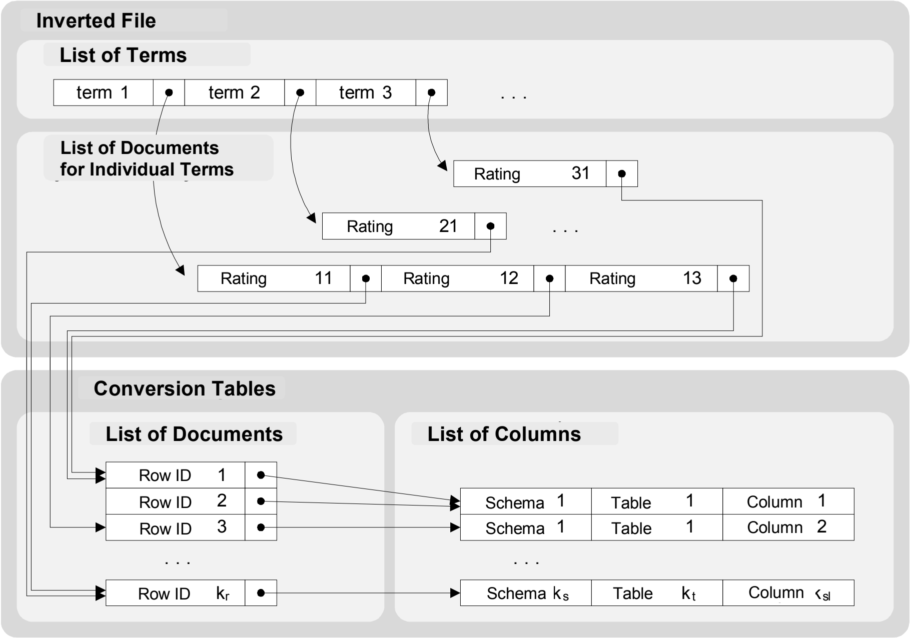
According to above stated requirements, it is necessary to consider each value stored in any of indexed columns as a separate textual document.
The Précis index creation process is shown on picture 2.



Picture 2. Précis index creation process

Each document obtains its unique identifier based on the table owner, table name, column name and row within the database. Document stored in binary format are filtered and converted to plaintext. Text is tokenized and converted to a set of terms together with number of their occurrences within the document. Filtering and tokenization uses functionality available in Oracle Media Text extension [3]. For each term its *term frequency* (TF) within a given document document and inverted document frequency (IDF) is computed and the standard TF*IDF formulae [4] is then used for term weight computation.

Logical structure of the Précis inverted index is shown on picture 3.



Picture 3. Logical structure of Précis index

Data are stored within BLOB records, divided to blocks of fixed size. Document lists short enough to not exceed one block, are stored directly in the block ordered by document ID. Longer lists are stored in form of non-redundant B-trees. Each BLOB record can contain blocks of the same type. It is possible to use more BLOB records containing ordered lists, providing that their block sizes differ each to other. Having more available blocksizes ranging from ones to hundreds items speeds up both searching and index synchronization.

First block in each BLOB – the header – contains metadata about the BLOB contents. The most of its content represents a beginning of free blocks list. The list continues after the last used block. During synchronization the free block list is stored in the core memory and is written back afterwards.

Position	Size	Description
0	2B	Number of used blocks in the record
2B	4B	Number of block with the rest of free records.
6B	4B	Number of blocks used for free blocks list.
10+	4B	Numbers of first 1000 free blocks within the BLOB.

Table 1: BLOB header structure

B-tree nodes in the BLOB records have similar format. Two bytes at the beginning of the node contain number of items stored within the node. Then follow pointers interlieved with item records. Item records contain the key and needed metadata. Metadata can either contain list of documents ordered by its numbers or its weights or it can contain lists of terms with lengths ranging from 1-6, 7-12, 13-24, respectively 25-64 characters. Details are shown in following tables.

B-tree purpose	**Field**	**Overal Size**	**Size**	**Decription**
Terms	Key	6, 12, 24, 64	6, 12, 24, 64	Term padded with trailing zero-bytes
	Metadata	12	2	Number of docs containing term
			2	Number of BLOB record with document list
			4	Number of block within the BLOB record containing list ordered by document number.
			4	Number of block within the BLOB record containing list ordered by term weigth. Zero, if the previous list is not stored as B-tree
Documents ordered by number	Key	4	4	Document number
	Metadata	2	2	Document weight
Documents ordered by weigth	Key	6	2	Document weight
			4	Document number
	Metadata	-	-	-

Table 2: B-tree item record structure

3.1 Programming Language

The application is split to two layers. Upper layer is written in PL/SQL language and contains procedures, monitoring database changes and the user interface. Lower layer is written in C++ and maintains internal index structures. Communication between layers is implemented thouigh OCI interface. The C++ performace is much higher than the alternative PL/SQL implementation. On the other hand, the invocation of procedures written in C++ from the PL/SQL code requires substantial overhead. The code was therefore written to minimize switches from PL/SQL to C++ as much as possible.

3.2 User Interface

The user wanting use Précis index has to have assigned role PRECISAPP containing the role CTXAPP that allows usage of Oracle Media Text features. To maintain indexes including foreign columns, the user has to posses additional privileges CREATE ANY TRIGGER, ADMINISTER DATABASE TRIGGER and SELECT on given TABLE. Due to security risks maintaining foreign columns is initially disallowed.

Index maintainance is done through package PRECIS with following interface.

Function/Procedure	Description
CREATE_INDEX	Creates index, auxiliary tables and needed triggers.
DROP_INDEX	Drops the index including all data.
DROP_INDEX_FORCE	The same as above, without existency checks.
ADD_COLUMN	Adds new table column to the index.
DROP_COLUMN	Removes the table column from the index.
SYNCHRONIZE	Indexes data added and/or changed from previous synchronization.
SET_AS_SINGLE_SCHEMA SET_AS_ MULTI_SCHEMA	Sets, if it is possible to add columns belonging to tables from other schemas.

Table 3: PRECIS package interface

3.3 Index Search

The search is accelerated through the domain index built upon column DOC_ID of table PRECIS$*index_name*$D. This table formally holds all documents, physically stored in their repective tables. The query searches data from this table or any view created over this table using operator PRECISSYS.CONTAINS. The system provides the view PRECIS$*index_name*$S that provides additional columns stored in internal index tables. The query would look like

```
SELECT S.*, PRECISSYS.SCORE(n)
  FROM PRECIS$index_name$S S
  WHERE PRECISSYS.CONTAINS(DOC_ID,Q,L,n) > Treshold
```

where *Q* holds Boolean expression with keywords, *L* means limit – maximal required number of hits and a *Treshold* keeps minimal required document rating. Last parameter *n* holds numerical identifier of operator within the SELECT statement and allows obtaining of assigned document rating using auxiliary operator SCORE with the corresponding identifier. Using Boolean expressions instead of simple keyword extends the original proposition. To formulate queries, users can use brackets, and logical operators AND "&", OR "|" and NOT "-". The structure of view PRECIS$*index_name*$S is described in following table. The column TEXT returns the beginning of the document and is present for testing purposes.

During evaluation, terms used in the query are identified in the index. If the length of the term exceeds given limit suffix_search_min_token_len, the index searches for all its rightwise extensions. Shorter terms are searched in their exact form.

Query evaluation is based on Paice model [5]. Index search then identifies all documents that have high weight assigned to at least one term used in the query. To identify them lists ordered by weight in descendant order by weight are used. The exact rating for these documents is then evaluated using lists ordered by document numbers.

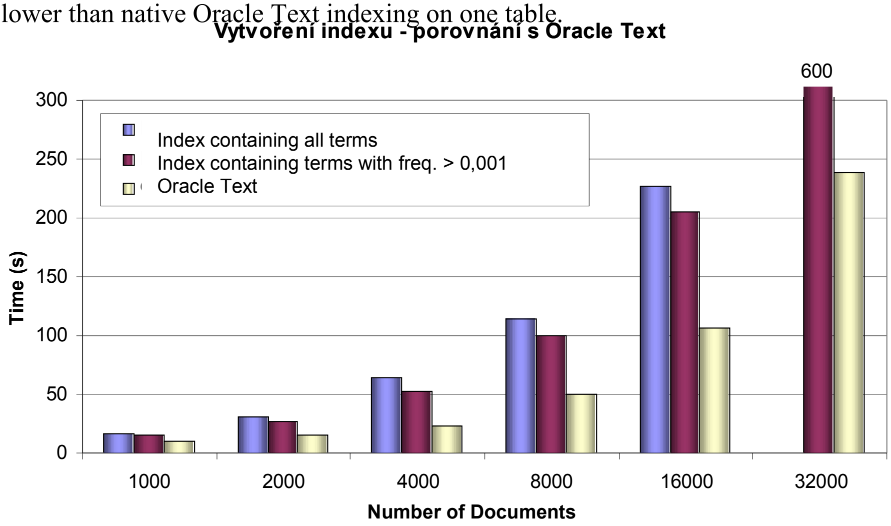
3.4 Index Data modification

Indexes are synchronized in batches. Changes are logged in the table PRECISSYS.PRECIS_PENDING. To keep track of changes Précis index uses two types of triggers. One of them tracks DML changes in tables and the second one watches for DROP TABLE and TRUNCATE TABLE statements. The INSERT OR UPDATE OR DELETE FOR EACH ROW trigger has to be created for each indexed table. Its name is PRECIS_*IndexID*_*TableID*_TRG. The trigger PRECIS_TBLDDL_TRG of the second type is created once in each schema containing at least one indexed column. Each document has assigned its unique identifier within the index, that can be translated to the quadruple (owner, table_name, column_name, rowid) through tables PRECIS$*index_name*$D and PRECIS$*index_name*$U. All BLOB records are stored in the table PRECIS$*index_name*$I. Parametrs of all precise indexes are stored in the table PRECISSYS.PRECIS_PARAMETER.

4 Performance tests

The graph on picture 4 shows result for index creation in comparison with *Oracle Multimedia Text*.

The time for 32000 documents increases significantly due to insufficient memory for buffers, so the data have to be transferred to a from the disk and the overall number of I/O operation was increased. In average, the index creation process is 2-3 times slower than native Oracle Text indexing on one table.



Picture 4. Précis index creation

The query evaluation was tested on collection containing 32000 documents from Wikipedia. Results represent execution times for queries Q_1 = “Astronomy”, Q_2 = “Information System”, Q_3 = “Pulp Fiction”, Q_4 = “Database Search Oracle MySQL”, Q_5 = “(Information Retrieval | Text Retrieval Systems)”, Q_6 = “Relational Database Index”.

	Q_1	Q_2	Q_3	Q_4	Q_5	Q_6
Rating computation	0.029	0.032	0.058	0.053	0.056	0.173
Data retrieval	0.052	0.163	0.205	0.259	0.241	0.430
Total	0.081	0.195	0.263	0.312	0.297	0.603

Table 4: Index search results

5 Conclusion

We proposed data structures suitable for Précis indexing and implemented it in the widespread Oracle database. The proposed implementation extends original idea by possibility to formulate queries using Boolean operators. The formulation of queries is familiar to Oracle application developers because of its similarity to Oracle Multimedia Text queries. Embeding most of document processing path provided by Oracle Media Text allows maintaining not only data in plain text columns, but also documents in different format stored in the database. The proposed data structures still provide quick searching of documents together with tolerable speed of indexing process.

References

1. Simitsis A. and col.: Précis: from unstructured keywords as queries to structured databases as answers. International Journal on Very Large Data Bases (VLDB Journal), Volume 17, Number 1, 2008, 117-149.
2. Čermák M.: Master Thesis: Index pro textové vyhledávání nad relačními daty, Charles University Prague, 2008. (in Czech language)
3. Oracle Text: Application Developer's Guide, 10g Release 2, 2005.
4. Salton, G. and M. J. McGill (1983). *Introduction to modern information retrieval.* McGraw-Hill. ISBN 0070544840.
5. Paice, C. P. (1984), *Soft Evaluation of Boolean Search Queries in Information Retrieval Systems*, Information Technology, Res. Dev. Applications, 3(1), 33-42

Fuzzy Approach to Landslide Susceptibility Zonation

Miloš Marjanović and Jan Caha

Palacky University in Olomouc, Faculty of Science
Tr. Svobody 26, 771 46 Olomouc, Czech Republic
milos.marjanovic01@upol.cz, jan.caha@klikni.cz

Abstract. The paper addresses a landslide-prone area on Fruška Gora Mt. in NW Serbia. It proposes a model of relative landslide susceptibility based on fuzzy sets. Having a variety of spatial attributes (proven statistically significant) at disposal, as well as present landslide inventory map, we conducted systematic analysis through (i) assigning fuzzy memberships to attribute categories, and (ii) combining the memberships by means of fuzzy operators. The performance defined by Area Under Curve parameter of the Receiver Operating Characteristics curve, led to preference of Frequency Ratio method for assigning memberships, and Fuzzy Gamma Operator for combining those memberships in 2-level experimenting configuration. Results are also well related with previous investigations with different approaches.

1 Introduction

Landslides and alike mass movements are one of the most widespread hazardous phenomena [1]. They seem to be among the top seven natural hazards, and advancing [19] in the world of growing needs for urbanization, land exploitation, and yet unstable climate conditions. Accordingly, there has been a significant ascent of interest in landslide assessment topics, resulting in more frequent multidisciplinary case studies and rising number of scholars per investigation [11].

Common notion of landslide hazard is broadly misinterpreted in relation to its conventional definition, which regards the hazard quantitatively as a function of frequency of hazardous phenomena over specified area or volume [23]. Nevertheless, even such precise scientific formulation is not entirely straightforward, since literal hazard assessment appears to be feasible only for the limited areas with excellent data coverage [4]. Entire range of problems is encountered in this framework, including the input data quality, lack of evidence on previous occurrences or triggering events, lack of consistent evaluation of the modeling results [4]. Therefore, most of the studies actually address landslide susceptibility as non-temporal variant of the landslide hazard, which evaluates the landsliding potential in the relative scale.

Practice of landslide zonation had been illustrated in versatile techniques in various case studies, yielding more or less reliable results depending on the complexity of the terrain and suitability of the approach [5]. Thereto, the principle assumption imply

V. Snášel, J. Pokorný, K. Richta (Eds.): Dateso 2011, pp. 181–195, ISBN 978-80-248-2391-1.

that future landslide occurrences stand in relation with the present ones [3], while central – multi-criteria modeling idea couples different input thematic data (geological, geomorphological, environmental maps), relate them to the referent map of present landslides, and processes a single output – hazard/susceptibility map. Techniques of relating referent landslide map with the inputs are numerous: heuristic (expert-based), deterministic (physically-based), statistical and probabilistic, artificial intelligence based (neural networks, decision trees, machine learning algorithms, data mining), fuzzy logic based, and so forth. All those equally face the non-linearity of the problem, and strong dependence on the referent landslide data, the entire input data feature space, for that matter.

Weather using ordinary fuzzy sets, or fuzzy measures, or even combining fuzzy with other statistical or classification approaches (Dampster-Shafer, K-means, Neural Networks) the ultimate advantage is seen in logics, which provides a substantial possibility for standardization of the analysis under the consideration [12]. Thus, the procedure tends to be repeatable, adjustable and reliable. When it comes to the landslide assessment analysis in particular, a number of researchers have applied fuzzy approach to handle the non-linearity, which is common in multi-criteria framework. Interestingly enough, Himalayan terrains were addressed in many investigations with fuzzy theory background, starting from standard fuzzy set approach [15], [21], [6], through combinations of neural-fuzzy [13] and risk-oriented fuzzy approach [14]. Most of these studies agreed that plausible susceptibility models could be obtained by applying advanced operators, with preference toward Cosine Amplitude method for obtaining memberships. Very similar conclusions with analogue methodology were inferred over Iranian case studies [22], and in Turkey [7], China [24] and so forth. The latter is also interesting in respect of harmonizing expert-based and fuzzy-driven solutions, inferring that one does not exclude another, but supports it. Finally, Regmi et al. [20] conducted one of the most consistent researches, where many different fuzzy configurations were put to test. Detailed elaboration of the choice of fuzzy operator type, optimal fitting of gamma operator as a method of preference, and some suggestions on handling multi-type landslide cases, can be found in this research. In addition, most of the researchers encourage the usage of the fuzzy method in other, similar or entirely different ambients, worldwide.

Herein we will concentrate on fuzzy logic approach, and compare results with some of earlier works that involved heuristic, statistical and machine learning techniques over the same area, using similar datasets. Thus, the primary objective is to investigate whether the fuzzy logic approach enhances the susceptibility model and to which extent. Optimization of the procedure, in accordance with the characteristics of the dataset, was also one of the foci, in order to reach the best performance of the model.

Organization of the paper goes as follows: in Chapter 2, a brief overview of all implemented techniques is presented; Chapter 3 follows with very basic description of study area; data acquisition and preparation is regarded in Chapter 4; results of susceptibility model and comparative analysis are presented and discussed in Chapter 5; Chapter 6. concludes the paper. Appendix 1 contains very detailed modeling parameters of attributes, which were used in the procedure.

All parametric calculations, were performed in MS Excel sheets, and spatial attributes were prepared and visualized with ArcGIS 9+ packages (ranged, calculated, cross-tabulated etc.), as well as final models.

2 Methods

2.1 Feature Selection

It is recommendable to filter the set for features which are of no relevance to the analysis, if nothing, for the sake of hardware and time expenditure, thus the filtering procedure is not to be overlooked. In parlance of the latter, a statistical significance tests needs to be run prior to the dataset utilization.

Chi-Squared statistic, parameter X^2, is a significance criterion, which relates the frequencies of observed independent variable instances φ_o within the dependent variable classes, and their expected frequencies φ_e, in the following fashion:

$$X^2 = \sum_{i=1}^{q} \sum_{j=1}^{n} \frac{(\phi_{o_{i,j}} - \phi_{e_{i,j}})^2}{\phi_{e_{i,j}}} , \qquad \textbf{(0)}$$

where q is the number of classes within a dependent variable, and n within the independent variable. In our case, the former represents landslide inventory classes, while the latter disclose the classes of a particular terrain attribute, since X^2 needs to be paired with every single attribute separately. The given terrain attribute disapproves the hypothesis of being statistically independent from the landslide inventory classes only if it exceeds the critical X^2 threshold, defined by the level of confidence (in respect with the normal distribution) and degrees of freedom (defined by reduced product of q and n, $(q\text{-}1)(n\text{-}1)$). In effect, this method reveals the relation of an attribute and the referent landslide inventory, but the ranking among multiple attributes is rather relative, primarily due to the measurement scale dependence of X^2 [2].

2.2 Fuzzy Set Theory

Concepts of fuzzy logic have a very long tradition in spatial analysis framework. Main purpose of fuzzy logic is to deal with vague information and with data that contain some kind of uncertainty [25]. When using fuzzy set theory or fuzzy logic, each object or statement is given value from interval <0,1> indicating its membership to the given set. Each object can be member of several sets with different membership values. This concept is very helpful for categorization of data and for decision making, because unlike Boolean logic it produces results valid with specific degree of truth. That helps with finding not only the perfect match for a given criteria, but rather shows how much each of possibilities meet given criteria. At some specific situations, when modeling physical geographical crisp sets, Boolean logic fail to provide correct and quality results because of natural substance of the phenomena at hand. In such cases, fuzzy set theory and fuzzy logic provides solutions for dealing with imprecise and vague data, which would be hard or even impossible to process by any other means.

2.3 Fuzzy Memberships

Membership value is determined by membership function. Membership function is a function that maps all given elements to interval of values <0,1>.

$$\mu_A : \mathrm{U} \rightarrow <0,1> \ , \qquad \mathbf{(0)}$$

where μ_A is a membership function, U is a set of elements. Then for each $x \in U$, $\mu_A(x)$ is membership value of the element x to the set A [25]. For purpose of this paper, we use two functions for computing the fuzzy membership values: Frequency Ratio and Cosine Amplitude.

Frequency Ratio gives proportion of landslide cells in the specific category for each of input layers. It can be described as ratio of relative frequency of landslide cells in a category (an attribute class) to the relative frequency of all landslide cells in the area:

$$\mathrm{FR} = \frac{N_{\mathrm{cell}}(L_i) / \mathrm{N}_{\mathrm{cell}}(C_i)}{N_{\mathrm{cell}}(L) / \mathrm{N}_{\mathrm{cell}}(C)} \ , \qquad \mathbf{(0)}$$

Where $N_{cell}(L_i)$ is the number of landslide cells in the category i, $N_{cell}(C_i)$ is the total number of cells in the category i, $N_{cell}(L)$ is total number of landslide cells and $N_{cell}(C)$ is the total number of cells. If the result is higher than 1 it shows higher density of landslide cells in the category then overall in the dataset. Results lower than 1, points to categories that have density of landslide cells lower then density in the dataset. To transform FR to membership values those outputs have to be normalized by dividing each FR by maximal FR in the given group of classes. Then the membership values are from the interval <0,1> and the higher the number is, the higher is the influence of this category on landslide occurrence.

Another method for determining the membership values of categories to the set of categories important for landslide occurrence is Cosine Amplitude method:

$$\mathrm{CA} = \frac{N_{\mathrm{cell}}(L_i)}{\sqrt{N_{\mathrm{cell}}(C_i) \cdot N_{\mathrm{cell}}(L)}} \ . \qquad \mathbf{(0)}$$

In this case, the membership value is calculated as ratio between number of landslide cells in the category and the square root of its product with the total number of landslide pixels in the dataset. Unlike FR the output values do not have to be normalized because they already fall in interval <0,1>.

2.4 Fuzzy Operators

Several fuzzy operators exist for combining membership functions. Best-known operators are AND and OR, but both of them suffer with problem that one of combined sets have significant impact on result of such combination while the other sets do not have such influence. In case of operator AND minimum of all values is the one that defines output and in case of OR operator it is the maximum value. Because of this reasons we use other operators such as Fuzzy Algebraic Product, Fuzzy Algebraic Sum, Gamma Operation and Weighted Average. All of them are described in detail [2] so only short review is given here.

In Fuzzy Algebraic Product and Fuzzy Algebraic Sum the outputs are defined as:

$$\mu_{\text{product}} = \prod_{i=1}^{n} \mu_i \ , \tag{0}$$

$$\mu_{\text{sum}} = 1 - \prod_{i=1}^{n} (1 - \mu_i) \ , \tag{0}$$

respectively, where n is number of membership function to be combined and μ_A is the i-th membership function. Fuzzy Algebraic Product tends to produce output function lower or equal to the lowest function given, while Fuzzy Algebraic Sum is complementary to the former, so it provides output function higher than all the inputs but never higher than 1.

Gamma Operation is defined by:

$$\mu_\gamma = (\mu_{\text{sum}})^\gamma \cdot (\mu_{\text{product}})^{1-\gamma} \ . \tag{0}$$

The exponent γ, which is a number from <0,1> interval, allows optimization of the membership combination. Setting it to the extremes of the interval give either Fuzzy Algebraic Sum (γ=1) or Fuzzy Algebraic Product (γ=0).

Weighted Average is defined as:

$$\mu_w = \frac{\sum_{i=1}^{n} w_i \cdot \mu_i}{\sum_{i=1}^{n} w_i} \ , \tag{0}$$

where w_i is a weight of membership function, indicating importance of the membership function on result and n is a number of membership functions to be combined. Weight system in this equation allows more interaction from user to the calculation, because it allows emphasis of certain values.

2.5 Performance Evaluation

Performance metrics involved Receiver Operating Characteristics (ROC), which is a cut-off independent performance estimator [9]. It involves contingency table inspection (derived by area cross-tabulations of attribute vs. landslide inventory). ROC values are created by plotting the cumulative True Positive Rates (TPR=sensitivity) versus False Positive Rate (FPR=1–specificity) for every model, resulting in a set of ROC curves. The performance is evaluated by the Area Under the Curve (AUC) relative to the entire plot area, so that an AUC equal to 1 has the best performance, while an AUC as low as 0.5 results in a very poor performance [10]. In addition, TPR is a good measurement of performance in the landslide assessment framework, since it takes into account instances that are not classified as landslides in the model but actually are landslides, which is more dangerous underestimation than false alarms.

3 Case Study

The study area encompasses the NW slopes of the Fruška Gora Mountain, in the vicinity of Novi Sad, Serbia. The site (N 45°09’20”, E 19°32’34” – N 45°12’25”, E 19°37’46”) spreads over approximately 100 km^2 of hilly landscape, but with interesting dynamics and an abundance of landslide occurrences. As judged in some previous investigations over this area [16], [17,], [18], the landslide process is chiefly governed by geological and morphological attributes, while the triggering mechanism could be assigned to excessive rainfall, but moderate seismic activity typical for this mountain, could also be an option.

4 Dataset

Dataset included geological, geo-morphometric, hydrological and environmental attributes, obtained from different resources, converted to raster grid format with 30 m cell resolution. It also included landslide inventory map.

- Geological data were assembled by using geological map 1 : 50 000, photo-geological map (Remote Sensing based interpretation of geological structures and geodynamic processes and forms) 1 : 50 000, and field survey data. For the purpose of this research, a segment of geological map was digitized and simplified to *geo-unit* attribute. Geological structures, which were used to make a *buffer geo-structures* were extracted from photogeological map. In addition, the *buffer geo-boundaries* was created by choosing only the boundaries between the units with significant difference in hydrogeological function.
- Model of the terrain surface was created from digitized contour maps at 1 : 25 000, first by calculating Triangulated Irregular Network (TIN) and then substituting it with the Digital Elevation Model (DEM) of 30 m resolution,

by means of TIN-to-raster data conversion. Given the terrain morphology, various geo-morphometric attributes were created as first order derivates of DEM: *aspect*, *elevation*, *slope angle*, *slope length*, *profile* and *planar curvature*.

- Hydrological attributes are represented by *topographic wetness index* (*TWI*) as the second order derivate of DEM, and *buffer stream* calculated after automatic generation of drainage pattern using DEM.
- *Land cover*, as an environmental attribute, was desirable in order to delineate deforested and cultivated areas as more convenient for the development of landslides than vegetated areas. The attribute was created by Landsat TM band ratioing (particularly red and near infrared bands, due to the authentic spectral behavior of vegetation). Several vegetation indices were considered, and Normalized Difference Vegetation Index (NDVI) seemed like the optimal solution, due to its simplicity and accuracy. Since the area of the interest is not very populated, urban influences were not considered. Classification of NDVI into land cover categories was semi-supervised, i.e. visual, but aided by K-means classification to four different entities (Appendix 1).
- *Landslide inventory map* was essential requirement to make a susceptibility assessment evaluated for performance. The map was created by extracting landslide forms from photogeological map. Subsequently, it was simplified to binary attribute (TRUE and FALSE landslide categories). It is important to mention constrains of such map, since it considered only earth slides [23] of rotational, translational and complex type, with two stages of the activity (dormant and active). This is understandable regarding the scale of the study (1 : 50 000) and the nature of the dominating landslide phenomena within the area of interest. According to this binary map, total of 10% of the area fall into landslide category (about 10 km^2).

Apparently, dataset involved continual numeric data, but categorical attributes as well. The methodological approach required ranging of continual attributes to categorical data, prior to their processing, and several solutions were regarded. Finally, ranging by means of Natural break cut-offs was the method of choice, which was applied to all continual attributes. Different continual attributes were ranged by appropriate number of intervals (Appendix 1), due to differences in pixel frequencies among attributes. In favor of selected approach of preparing the data, feature selection parameter proved that all attributes had statistical dependence to referent landslide inventory, having the values significantly higher than critical (Appendix 1).

5 Results and Discussion

Given the categorized (ranged) raster attributes and the referent landslide inventory map, we first calculated the memberships of each category in each attribute. Two parallel variants of the experiment were driven: EXPERIMENT 1 used Cosine Amplitude, while EXPERIMENT 2 used Frequency Ratio to obtain the memberships.

Both experiments had exactly the same course, thus the following manipulations took place in each.

5.1 Susceptibility Model

In order to combine memberships by different operators we undertook a small intervention to exclude too many extreme membership values (0 and 1) by replacing them with close approximations (0.0001 and 0.9999). We proposed 2-level fuzzy combinations based on *a priory* knowledge of the phenomena (Fig. 1), i.e. the pairs of attributes of similar origin were grouped together. Continual Susceptibility Model was obtained after the second level combination. The final susceptibility model was generated by ranging the continual values into five standard categories of relative susceptibility: Very Low – VL, Low – L, Moderate – M, High – H, Very High - VH [8]. Regarding the distribution of the pixels in Continual Susceptibility Model, it was justifiable to adopt the quantile interval cut-offs for afore mentioned categorization. Only the highest susceptibility class VH was regarded for performance evaluation (AUC) against the referent landslide inventory (Table 1). This was instructed by the fact that determined landslides should be marked as a priority zone (preferably as VH class).

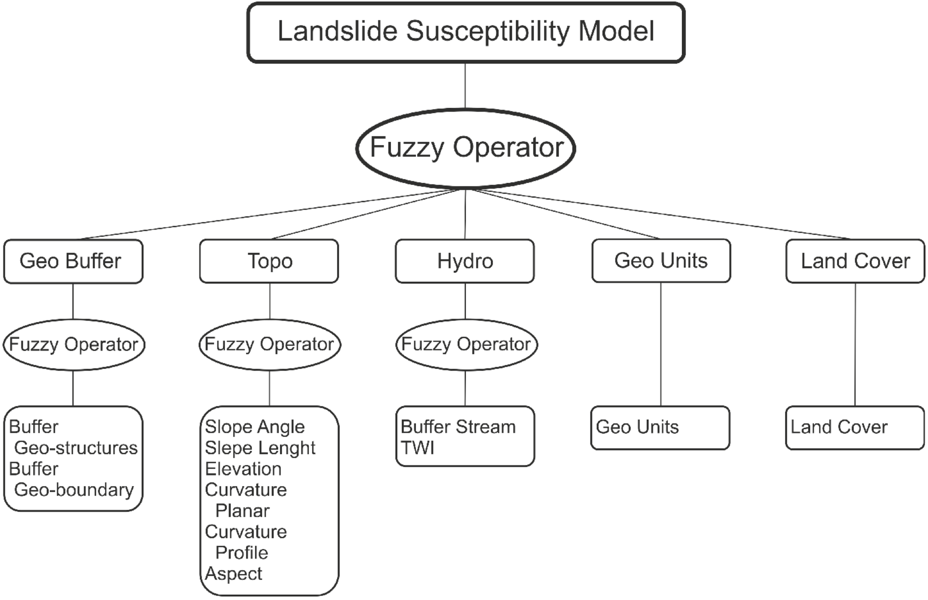

Landslide Susceptibility Model
Fuzzy Operator
Geo Buffer
Topo
Hydro
Geo Units
Land Cover
Fuzzy Operator
Fuzzy Operator
Fuzzy Operator
Buffer Geo-structures Buffer Geo-boundary
Slope Angle Slepe Lenght Elevation Curvature Planar Curvature Profile Aspect
Buffer Stream TWI
Geo Units
Land Cover

Fig. 1. Flowchart of the experiment configuration.

To remain consistent, we kept the same type of the operator at both combination levels. Initial results in both experiments gave preference to Fuzzy Gamma Operator, so we directed further fitting toward optimization of parameter γ. Cases of γ=0 (Fuzzy Product) and γ=1 (Fuzzy Sum) were already regarded, so we tested several choices within that interval (0.25, 0.5, 0.75). It turned that the best performance (AUC) was achieved by γ=0.5, making it a parameter of choice for our final susceptibility model. Finally, EXPERIMENT 2 gave slightly better performance over EXPERIMENT 1, meaning that Frequency Ratio could be preferred over Cosine Amplitude for assigning memberships.

Table 1. Performance evaluation of different fuzzy experiments configurations (1-Cosine Amplitude, 2-Frequency Ratio memberships), and other landslide susceptibility models (shaded)

Model	**AUC**	**TPR**
EXPERIMENT 1 (Weighted Average)	0.65	0.37
EXPERIMENT 1 (Gamma Operator, γ=0.5)	0.70	0.53
EXPERIMENT 2 (Weighted Average)	0.71	0.56
EXPERIMENT 2 (Gamma Operator, γ=0.5)	0.72	0.58
AHP	0.67	0.48
CP	0.72	0.60
SVM	0.85	0.77

Distribution of relative susceptibility classes goes as follows: VL – 53%, L – 14%, M – 12%, H – 11%, and VH – 10%. Dominance of the VL class characterizes the terrain as mostly stable, while similarly as in the referent inventory map, the most adverse zones occupy about 10% of the area. Furthermore, a majority of the actual landslide instances fall into the VH and H classes (37% and 23% of all landslides, respectively), while M, L and VL classes occupy mostly non-landslide instances (75% of non-landslide instances in total for all three classes).

Highest overall performance in EXPERIMENT 2 (AUC=0.72) could be acknowledged as plausible, which is also supported visually (Fig. 2a-b), since VH class corresponds very well with the spatial trends of landslide scarps. Apparent influence of intermediate layer *Geo Buffer* caused several outliers by underestimating some landslide scarps. A considerable drawback is relatively low TPR in both experiments (Table 1) which is inconvenient for any hazard-related analysis, since the model tends to underestimate actual landslide instances (claiming class other than VH for an actual landslide instance). However, the actual performance is somewhat better, since we regarded only VH class for cross-tabulation. Thus, H or even M class could be fair replacements for VH class, as they buffer-out around it, which if included in cross-tabulation might reduce the number of False Negatives, thus increasing TPR.

5.2 Comparison

In order to determine the true practicality of our results, we related proposed model to other available results including: Analytical Hierarchy Process (AHP) model [16], Conditional Probability (CP) model [18], and machine learning with Support Vector Machine (SVM) model [17]. When comparing the best fuzzy-based result with the other models the same policy of comparing only VH class holds, due to compatibility issue. Namely, some of the comparison models, such as SVM, are discrete in their nature and cannot follow (standardized) relative susceptibility categorization (VL–VH).

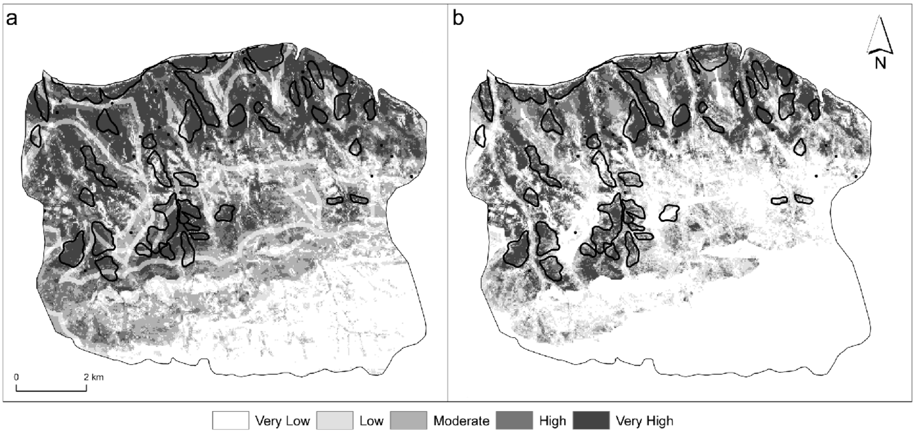


Fig. 2. Landslide susceptibility models based on EXPERIMENT 1 (CA, γ=0.5) **a)**, and EXPERIMENT 2 (FR, γ=0.5) **b)**. Bold contours outline the landslide scarps from the landslide inventory. Legend depicts relative susceptibility classes.

Expectedly, SVM approach outperformed fuzzy-based models by far (Table 1). Ease of handling continual and categorical data most likely enables such dominance of SVM model over other results. On the other hand, fuzzy approach turned practically as successful as statistical one (CP model), but with more subjectivity involved in the modeling procedure (in ranging the input intervals, but also in selecting the operators and numbers of combination levels). It outperformed AHP model, not as much in the overall performance (AUC) as in considerably higher TPR, giving itself a slight preference for safer assessment (Table 1).

6 Conclusion

In present paper, we regarded fuzzy set approach in the landslide susceptibility framework, having different input attributes and referent landslide inventory at disposal. Subjectivity in ranging input attributes was inevitable, due to incapability of the approach to handle continual numerical variables (in the stage of assigning memberships). Another subjective intervention regarded proposing the number of levels for fuzzy combination, and grouping the attributes with similar origin at level 1. We proposed two configurations of generating memberships of input attribute categories, EXPERIMENT 1 (CA) and EXPERIMENT 2 (FR), and led further optimization toward the choice of fuzzy operators for combination task. The best performance was reached with Fuzzy Gamma Operator with γ=0.5. The resulting Landslide Susceptibility Model turns plausible, and seems improved when compared to some previous models designed for the same study area, particularly heuristic one.

Further refinement, left for the future work, should involve combining of fuzzy approach with some other techniques. The latter primarily address merging with heuristic expert decisions, while fuzzyfication in machine learning approach is also to be challenged. Another improvement could be recognized in reducing the subjectivity in experiment design, and configure the experiment structure on statistical basis or information theory basis.

To conclude, our research came up with suitable model, while the procedure remained simple, semi-automated and re-operable in GIS environment. The resulting map could serve preliminary levels of risk or disaster management, landscape (regional) planning, route selection, insurance management and so forth.

Acknowledgement

This work was supported by the Czech Science Foundation (Grant No. 205/09/079).

References

1. Aleotti, P., Chowdhury, R.: Landslide hazard assessment: summary review and new perspectives. Bulletin of Engineering Geology and the Environment, Vol. 58, Springer-Verlag (1999) 21-44
2. Bonham-Carter, G.: Geographic information system for geosciences – Modeling with GIS. Pergamon, Oxford (1994)
3. Carrara A., Guzzetti F., Galli M., Cardinali M., Reichenbach P.: Predicting regional landslide hazard. In: Proceedings of 1st European Congress on Regional Geological Cartography and Information Systems, Bologna, 13-16 June (1994) 50-52
4. Carrara, A., Pike, R.: GIS technology and models for assessing landslide hazard and risk. Geomorphology Vol. 94, Elsevier (2008) 257-260
5. Chacón, J., Irigaray, C., Fernández, T., El Hamdouni, R.: Engineering geology maps: landslides and geographical information systems. Bulletin of Engineering Geology and the Environment, Vol. 65, Springer-Verlag (2006) 341-411
6. Chamaptiray ray, P. K., Dimri, S., Lakhera, R. C., Sati, S.: Fuzzy-based method of landslide hazard assessment in active seismic zone of Himalaya. Landslides, Vol. 4.Springer-Verlag (2006) 101-111
7. Ercanoglu, M., Gokceoglu, C.: Use of fuzzy relation to produce landslide susceptibility map of a landslide prone area (West Black Sea Region, Turkey). Engineering Geology, Vol. ,75, Elsevier (2004) 229-250
8. Fernández, T., Irigaray, C., El Hamdouni, R., Chacón, J.: Methodology for Landslide Susceptibility Mapping by Means of a GIS. Aplication to the Contraviesa Area (Granada, Spain).Natural Hazards, Vol. 30, Kluwer Academic Publishers (2003) 297-308
9. Fielding, A.H., Bell, J.F.: A review of methods for the assessment of prediction errors in conservation presence/absence models. Environmental Conservation, Vol. 24/1, Cambridge Journals (1997) 38-49

10. Frattini, P., Crosta, G., Carrara, A.: Techniques for evaluating performance of landslide susceptibility models. Engineering Geology, Vol. 111, Elsevier (2010) 62-72
11. Gokceoglu, C., Sezer, E.: A statistical assessment on international landslide literature (1945-2008). Landslides, Vol. 6, Springer-Verlag (2009) 345-351
12. Jiang, H., Eastman, J. R.: Application of fuzzy measures in multi-criteria evaluation in GIS. Int. J. Geographical Science, Vol. 14. Taylor & Francis (2000) 173-184
13. Kanungo, D. P., Arora, M. K., Sarkar, S., Gupta, R. P.: A comparative study of convetional, ANN black box, fuzzy and combined neural and fuzzy weighting procedures for landslide susceptibility zonation in Darjeeling Himalayas. Engineering Geology. Elsevier (2006) 347-366
14. Kanungo, D. P., Arora, M. K., Gupta, R. P., Sarkar, S.: Landslide risk assessment using concepts of danger pixels and fuzzy set theory in Darjeeling Himalayas. Landslides, Vol. 5. Springer-Verlag (2008) 407-416
15. Kanungo, D. P., Arora, M. K., Sarkar, S., Gupta, R. P.: A fuzzy set based approach for integration of thematic maps for landslide susceptibility zonation. Georisk, Vol. 3. Taylor& Francis (2009) 30-43
16. Marjanović, M.: Landslide susceptibility modeling: A case study on Fruška Gora Mountain, Serbia. Geomorphologia Slovaca et Bohemica, Vol. 9/1. Slovac Academy of Science (2009) 29 - 42
17. Marjanović, M., Bajat, B., Kovačević, M.: Landslide susceptibility assessment with machine learning algorithms. In: Proceedings of International Conference on Intelligent Networking and Collaborative Systems, INCoS 2009, IEEE (2009) 273-278
18. Marjanović, M.: Regional scale landslide susceptibility analysis using different GIS-based approaches. In: Williams et al. (eds): Geologically Active, Taylor & Francis Group, London (2010) 435-442
19. Nadim, F., Kjekstad, O., Peduzzi. P., Herold, C., Jaedicke, C.: Global landslide and avalanche hotspots. Landslides, Vol. 3, Springer-Verlag (2006)159-173
20. Regmi, N. R., Giardino, J. R., Vitek, J. D.: Assessing susceptibility to landslides: Using models to understand observed changes in slopes. Geomorphology. Elsevier (2010) 25-38
21. Srivastava, V., Srivastava, H. B., Lakhera, R. C.: Fuzzy gamma based geomatic modeling for landslide hazard susceptibility in a part of Tons river valley, northwest Himalaya, India. Geomatics, Natural Hazards and Risk, Vol. 1. Taylor & Francis (2010) 225-242
22. Tangestani, M. H., Landslide susceptibility mapping using the fuzzy gamma approach in a GIS, Kakan catchment area, southwest Iran. Australian Journal of Earth Sciences (2004) 439-450
23. Varnes, D.J.: Landslide hazard zonation: A review of Principles and Practice. International Association for Engineering Geology, Paris (1984)
24. Wang, W., Xie, C. Du, X.: Landslides susceptibility mapping in Guizhou province based on fuzzy sets theory. Mining Science and Technology. Elsevier (2009) 399-404
25. Zadeh, L. A.: Fuzzy sets. Information and Control, Vol. 8/3, Elsevier (1965) 338-353

Apendix 1– Table of Attributes

Input attributes, their class memberships in EXPERIMENT 1 configuration (μ_{FR}) and EXPERIMENT 2 configuration (μ_{CA}), and their statistical dependence (X^2) on landslide inventory (dependent variable)

attribute name (type, group)	μ_{FR}	μ_{CA}	X^2
categories			($X^2_{critical}$)
buffer geo-structure (continual, geo-buffer)			1949.6
0 - 134	0.051	0.781	(27.9)
134 - 276	0.092	0.770	
276 - 426	0.141	0.805	
426 - 582	0.260	1	
582 - 755	0.261	0.812	
755 - 942	0.178	0.445	
942 - 1159	0.024	0.077	
1159 - 1418	0	0	
1418 - 1758	0.262	0.197	
1758 – 2305 m	1	0.568	
buffer geo-boundary (continual, geo-buffer)			306.3
0 - 94	0.275	1	(27.9)
94 - 218	0.038	0.630	
218 - 342	0.107	0.499	
342 - 458	0	0.372	
458 - 589	0.040	0.295	
589 - 726	0.484	0.429	
726 - 878	0.579	0.313	
878 - 1050	1	0.332	
1050 - 1244	0.458	0.105	
1244 – 1749 m	0.682	0	
buffer stream (continual, hydro)			2381.6
0 - 94	0.550	0.643	(27.9)
94 - 212	0.900	1	
212 - 324	0.780	0.809	
324 - 432	0.453	0.431	
432 - 543	0.346	0.305	
543 - 660	0.218	0.165	
660 - 797	0	0	
797 - 966	0.024	0.002	
966 - 1173	0.362	0.100	
1173 – 1542 m	1	0.214	
TWI (continual, hydro)			4947,6
7.5 - 9.3	0	0	(26.1)
9.3 - 10.3	0.172	0.261	
10.3 - 11.4	0.696	1	
11.4 - 12.8	1	0.955	
12.8 - 14.3	0.811	0.575	
14.3 - 16.2	0.821	0.442	
16.2 - 18.3	0.781	0.320	
18.3 - 20.8	0.506	0.176	
20.8 - 22.5	0.131	0.079	

aspect (categorical, topo)			1091.0
flat	0	0	(26.1)
N	0.594	0.701	
NE	0.552	0.688	
E	1	1	
SE	0.889	0.490	
S	0.278	0.140	
SW	0.645	0.638	
W	0.494	0.571	
NW	0.414	0.433	
elevation (continual, topo)			7515.7
78 - 102	0.660	0.619	(27.9)
102 - 138	1	1	
138 - 173	0.828	0.838	
173 - 209	0.530	0.499	
209 - 248	0.158	0.147	
248 - 287	0.141	0.118	
287 - 329	0.018	0.013	
329 - 376	0	0	
376 - 426	0	0	
426 – 540 m	0	0	
slope angle (continual, topo)			4453.1
0 - 4.2	0.300	0.243	(18.5)
4.2 - 9.5	1	1	
9.5 - 14.8	0.473	0.403	
14.8 - 21.1	0.119	0.086	
21.1 - 40.1º	0	0	
slope length (continual, topo)			1346.8
0 - 60	0.435	1	(27.9)
60 - 181	0.591	0.964	
181 - 353	0.937	0.960	
353 - 602	1	0.667	
602 - 981	0.650	0.261	
981 - 1506	0.301	0.080	
1506 - 2196	0.178	0.033	
2196 - 3094	0.427	0.061	
3094 - 4392	0.187	0.019	
4392 – 6499 m	0	0	
plan curvature (continual, topo)			989.4
concave	0	0	(18.5)
-	0.657	0.333	
flat	1	1	
-	0.626	0.419	
convex	0.149	0.059	
profile curvature (continual, topo)			1214.0
concave	0	0.009	(18.5)
-	0.414	0.287	
flat	1	1	
-	0.741	0.366	
convex	0.081	0	
geo-units (categorical, geo-units)			8319.3
al' - Danube's inundation plane	0.100	0.099	(29.6)
al - aluvium	0.211		
dl - deluvium cover	0.807	1	
t - terrace sediments	1	0.785	

l - loess	0.338	0.334	
Pl - clay	0.858	0.847	
M_2 - marlstone	0.133	0.083	
M_1 - limestone, sandstone	0.469	0.880	
Se - ultra-mafic rocks	0	0	
J - limestone	0	0	
Pz - schists	0.002	0.003	
land cover (categorical, land cover)			6316.2 (16.2)
water	0	0	
arable land	1	1	
grass land	0.992	0.852	
forest	0.132	0.168	

Fractal Dimension Calculation for CORINE Land-Cover Evaluation in GIS – A Case Study

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

[1] Department of Geoinformatics, Faculty of Science, Palacký University in Olomouc
Tř. Svobody 26, 771 46 Olomouc, Czech Republic
[2] Department of Mathematical Analysis and Applied Mathematics, Faculty of Science
Palacký University in Olomouc, 17. listopadu 12, 771 46 Olomouc, Czech Republic
vit.paszto@gmail.com, lukas.marek@upol.cz, pavel.tucek@upol.cz
www.geoinformatics.upol.cz

Abstract. Together with rapid development in GI science recent decades, the fractal geometry represents a powerful tool for various geographic analyses and studies. This study points the land-cover areas with extreme values of fractal dimension in Olomouc region. This leads, together with consequent statistical analyses, to result that according to fractal dimension it is possible to distinguish (or at least to assume) the origin of areas. General fractal calculation method is used in the case study. Statistical methods are also applied to test mean values of land-cover areas fractal dimension (Student's t-test and analysis of variance). Using non-integer, fractal dimension, one can analyze complexity of the shape, explore underlying geographic processes and analyze various geographic phenomena in a new and innovative way.

Keywords: fractal geometry, GIS, land-cover, fractal dimension, geocomputation, shape metrics.

1 Introduction

When Weierstrass's continuous nonwhere-differentiable curve appeared in 1875, it was called by other mathematicians as "regrettable evil" and these types of object were known as mathematical "monsters" [8, 18]. Nobody imagined that fundamentals of fractal geometry were just established. However, since Mandelbrot's published its basics in [13], fractal geometry and fractal dimension (non-integer dimension, e.g. 1.32 D) is well known as a valuable tool for describing the shape of objects. It gained great popularity in geosciences [1, 7] (among other disciplines), where the measures of object's shape are essential.

Complex and detailed information about fractal geometry is in [8, 11, 14, 15, 18, 19]. Books provide the broad view of the underlying notions behind fractals and, in addition, show how fractals and chaos theory relate to each other as well as to natural phenomena. Especially introduction of fractals to the reader with the explicit link to natural sciences, such as ecology, geography (demography), physical geography, spatio-temporal analyses and others is in [8]. Some papers concerning topics investigated in this paper (land-use/land-cover pattern) were published yet, e.g. Batty and Longley's book [1] as pioneer work. Many other studies, such as [2, 5, 6, 16, 24, 26], applied different fractal methods for description of city morphology and connected issues. Fractal analyses applied especially on land-use/land-cover pattern are described as well, such as in [5, 9, 10, 17, 22, 27, 28].

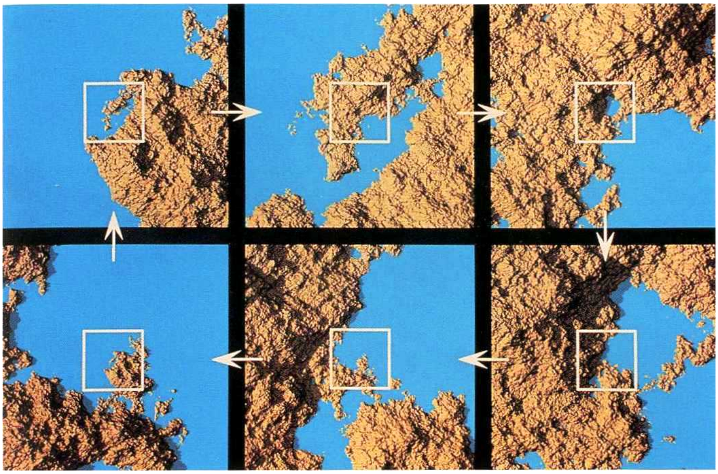


Fig. 1. Example of fractal coast and scale-invariance principle (in six steps/scales) [18].

One of the major principles in chaos theory and descriptive fractal geometry is self-similarity and self-affinity. The most theoretical fractal objects, such as Mandelbrot set, are self-similar – this means that any part of the object is exactly similar to the whole. But these types of fractals are rarely used to approximate objects or shapes from the real world. And thus, another type of fractals is suitable for real-world object description – self-affine ones. These fractals are in fact self-similar too, but transformed via affine transformation (e.g. translation, rotation, scaling, shear mapping) of the whole or the part of fractal object [1, 8, 11, 15, 18, 19]. This observation is closely related to scale-invariance (Fig. 1), which means that object has same properties in any scale, in any detail. In other words, if characteristics of some fractal object are known in certain scale, it is possible to anticipate these characteristics of another fractal object in different scale. The very typical example of this object is land-cover and/or urban forms with theirs dynamics.

Concept of fractality was described in detail in many publications [4, 7, 15, 18, 23]. Fractal dimension is a measure of complexity of the shape, based on irregularity, scale dependency and self-similarity of objects [2]. The basic property of all fractal structures is their dimension. Although there is no exact definition of fractals, the publicly accepted one, coming from Mandelbrot himself: "A fractal is by definition a set for which the Hausdorff-Besicovitch dimension strictly exceeds the topological dimension" [15]. Hausdorff-Besicovith dimension is therefore a number, which describes the complexity of an object and its value is non-integer. The bigger the value of Hausdorff-Besicovitch dimension, the more complex the shape of object and the more fills the space. In sense of Euclidean geometry, dimension is 1 for straight line, 2 for circle or square and 3 for cube or sphere, all. For real objects in plane, Hausdorff-Besicovitch dimension (fractal dimension) has values greater than 1 and less than 2. It obvious that Euclidean, integer, dimension is extreme case of fractal, non-integer, dimension. So it is claimed that regions with regular and less complex shape has lower fractal dimension (approaching to 1) and vice versa – the more irregular and complex shapes, the higher fractal dimension (approaching to 2). Values of fractal dimension of land-cover regions vary between 1 and 2 because of the fact that area represented in the plane space without vertical extend is in fact enclosed curves. And fractal dimension of curves lies between 1 and 2.

As depicted in mentioned publications, fractals provide tool for better understanding the shape of given object. Furthermore, fractal geometry brings very effective apparatus to measure object’s dimension and shape metrics in order to supply or even substitute other measurable characteristics of the object.

Next paragraphs do not intent to completely identify socio-economical, demographical and geographical aspects of land-cover current state in Olomouc region. The case study demonstrates the opportunity and power of fractal analyses of geographical data. Particularly, objectives are: land-cover pattern and its geometric representation in GIS. Land-cover pattern fractal analyses, among others, identify areas with maximal and minimal fractal dimension to evaluate complexity of such areas.

2 Methods, Data and Study Region

There is a number of methods for estimating fractal dimension and as [20] shows, results obtained by different methods often differ significantly. Also not only the method itself, but also the software, which calculates the fractal dimension may contribute to the differences [20]. In this case, ESRI ArcGIS 9.x software was used.

It has to be mentioned that in the case study statistical testing was accomplished. For this purpose, R-project statistical software was used. Because of the well-known formulas and characteristics, detailed description of the methods is not stated, excluding the program code in R-project environment. The methods were, namely, analysis of variance (hereafter as ANOVA) and Tukey's honest significant difference test and t-test.

2.1 General Calculation of Fractal Dimension

As mentioned above, there exists a plenty of methods to calculate fractal dimension [3, 6, 8, 18, 25]. For this purpose, one of the most general fractal dimension formula is used:

$$D = \frac{2 \cdot \log P}{\log A}\ . \tag{$\mathbf{0}$}$$

Where P is the perimeter of the space being studied at a particular length scale, and A is the two-dimensional area of the space under investigation [26]. Formula (**1**) was used to calculate fractal dimension for land-cover regions classified by Level 1 of the hierarchy. Formula (**1**) can be easily computable directly within GIS, thus ESRI ArcGIS 9.x was used in order to obtain fractal dimension values.

2.2 Statistical Evaluation of Fractal Dimension

After the fractal dimensions of land-cover shapes were calculated, the statistical evaluation was done. Firstly, the differences of fractal dimension among objects of various origins were testing using ANOVA. A null hypothesis, stated as ”there are no differences among mean values of the classes”, was formulated. In the case that null hypothesis was denied, the differences among groups were statistically significant and Tukey's honest significant difference test (hereafter as TukeyHSD) was performed. Although TukeyHSD is weaker test than ANOVA [21], it allowed multiple comparison procedure and statistical testing for finding particular differences among class couples. Thus, it could be helpful in the evaluation of fractal analysis [12]. If the criterion of TukeyHSD was on the boundary between a distinction and non-distinction, the t-test was also performed.

Example of a program commands used in R-project for an analysis of variance:

```
setwd("C:/Data/")
fd=read.csv2("fd.txt")
  anova<-aov(fd[,1]~fd[,2])
  summary(anova)
  TukeyHSD(anova)
  plot(TukeyHSD(anova))
```

2.3 Data and Study Region

For the case study, territory of Olomouc region was used (Fig. 2). Its area is approximately 804 km^2 and every single type of LEVEL1 land-cover classification is represented. It is necessary to note that CORINE Land-Cover dataset from year 2000 was examined. Olomouc region is mainly covered by the agricultural areas, but the north-east part is almost completely covered by forests, because of military area occurrence. Despite this fact, Olomouc region is the most typically agricultural region with a great number of dispersed villages (Fig. 3).

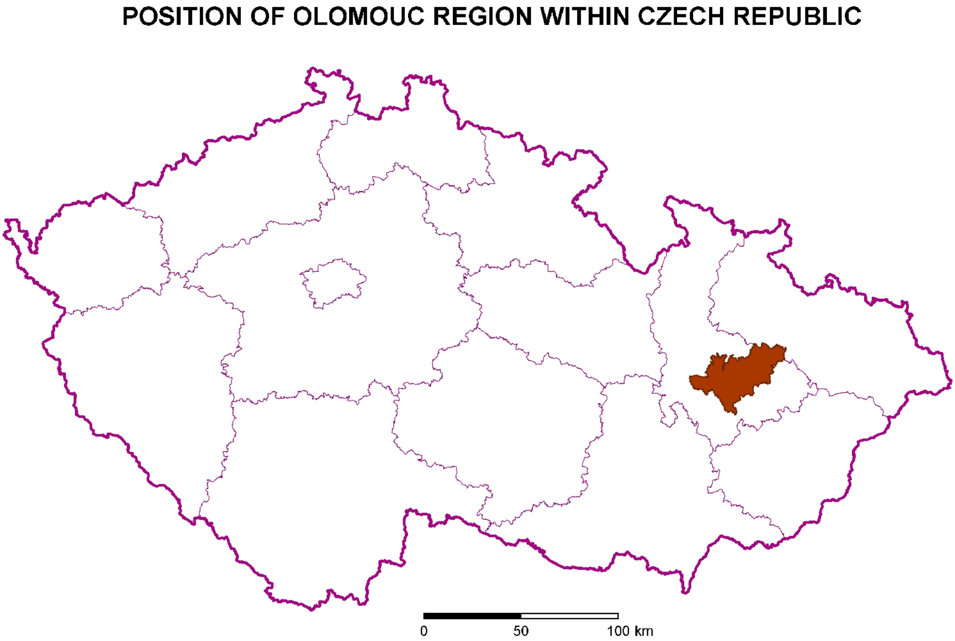


Fig. 2. Position of Olomouc region within Czech Republic (brown filled area).

3 Case Study: Fractal Analysis of Land-Cover within Olomouc Region

Visualization of land-cover in Olomouc region, which has fractal structure typical for landscape, is shown in Fig. 3. Areas with maximal and minimal fractal dimension, both for artificial areas and natural areas, are also outlined. From *Artificial areas*, the maximal fractal dimension (*D=1.393*) has town Hlubočky (Mariánské Údolí) and the minimal value of fractal dimension has part of Bystrovany municipality (*D=1.220*). In the first case, the maximal fractal dimension is caused by the topography of the town. Hlubočky (Mariánské Údolí) was built in steep valley on both sides of the river and thus is forced to follow highly irregular topography, which results into observed fractal dimension.

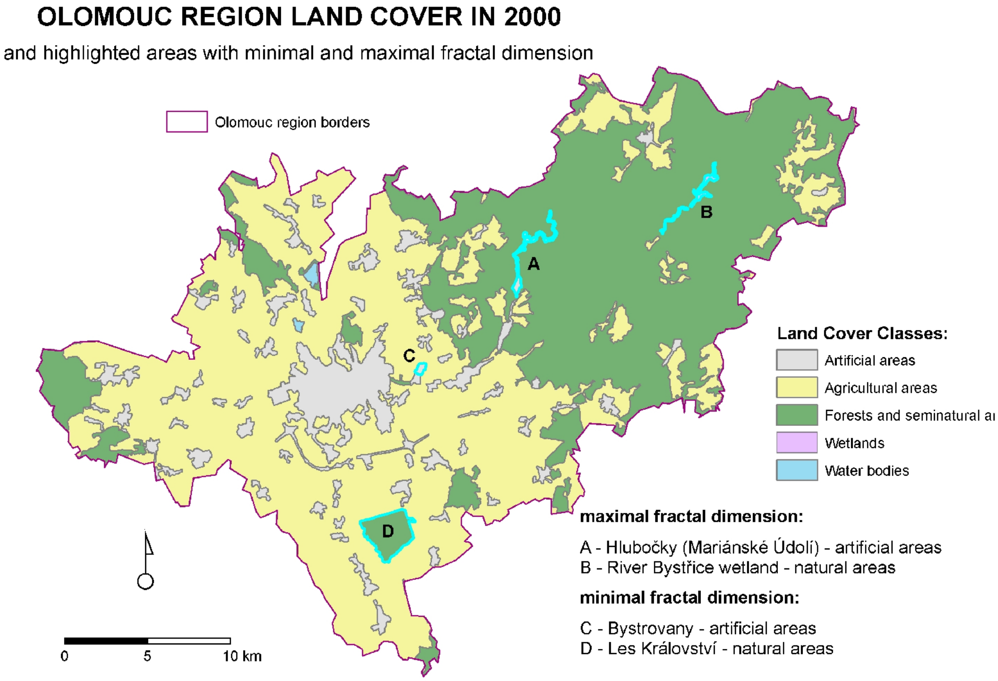

OLOMOUC REGION LAND COVER IN 2000
and highlighted areas with minimal and maximal fractal dimension
Olomouc region borders
A
B
C
D
Land Cover Classes:
Artificial areas
Agricultural areas
Forests and seminatural ai
Wetlands
Water bodies
maximal fractal dimension:
A - Hlubočky (Mariánské Údolí) - artificial areas
B - River Bystřice wetland - natural areas
minimal fractal dimension:
C - Bystrovany - artificial areas
D - Les Království - natural areas
0
5
10 km

Fig. 3. Olomouc region land-cover in 2000 and highlighted areas with minimal and maximal fractal dimension.

On the contrary, part of Bystrovany municipality represents distinct regular shape – almost square. There were no landscape borders or limitations when the settlement was built and regular fabrication of the build-up area (agricultural facility) was, probably, the most logical one. From natural areas, maximal fractal dimension has the *Wetland area* of Bystřice river (*D=1.396*), which is part of highlands with almost intact landscape. Very regular shape has forest southern from Olomouc called Les Království and its fractal dimension (*D=1.193*) corresponds with that fact.

At last, join of all areas within class was accomplished and overall fractal dimension calculated. Results are shown in Table 1.

Table 1. Overall fractal dimension of particular land-cover classes in Olomouc region.

Class index	Land-Cover Class	Fractal Dimension
A	Artificial areas	1.438574
B	Agricultural areas	1.385772
C	Forests and seminatural areas	1.350355
D	Wetlands	1.395799
E	Water bodies	1.263722

It is clear from Table 1 that highest fractal dimension have *Artificial areas*, which represents in the very most cases man-made build-up areas (villages, towns, various facilities). Although knowledge how to plan and build up the settlement more properly was known long ago, urban sprawl emerged and has great influence on the irregular shape of artificial areas. *Wetlands* are very specific class, which are fully determined by natural processes and its fractal dimension is the highest among natural areas. On the other hand, *Water bodies* have the lowest overall fractal dimension. It is necessary to note that line objects, which would fall into this class (rivers, streams, channels, etc.), are excluded due to CORINE classification methodology. And that is why the water bodies have this overall fractal dimension – only man-made or man-regulated water bodies were identified by the classification process and consequently analyzed.

To objectively prove the significant statistical difference among the land-cover classes, the ANOVA was used. Before that, Shapiro-Wilk test (W=0.98) was performed to check up the normality of data. Normality was confirmed and ANOVA could be used. It was then proven that mean fractal dimension values are significantly different among areas of the various origin and thus the classes are different too. One can then claim that classes (Table 1) originate from diverse processes. To acquire more detailed information, TukeyHSD was performed. This test allowed multiple comparisons among classes and identified particular statistically significant differences. TukeyHSD had two main outputs, tabular output, which mainly contained p-value of difference between two classes, and also the graphical output, which described the differences and is self-explanatory.

The graphical output (Fig. 4) clearly shows classes with higher variability or which are more similar in chosen characteristics. Based on the TukeyHSD, it was possible to state that *Wetlands* (D) were the most different from all classes. Furthermore, the second class, which could have been recognizable is *Forests and seminatural areas* (C), which were on the boundary of a distinction with regard to *Artificial areas* (A) and *Agricultural areas* (B). T-tests proved that class *Forests and seminatural areas* really differed from both, *Artificial areas* (p-value=0.006) and *Agricultural areas* (p-value=0.015). Based on previous findings, one can claim that by calculating the fractal dimension of land-cover areas and consequence statistics it is possible to study, evaluate and interpret the processes lying underneath the current land-cover appearance.

According to the CORINE Land-Cover classification system and acquisition of the dataset in reference scale 1: 100.000, influence of generalization on the fractal analysis needs to be taken into account. The more generalized areas in land-cover classes, the more regular their shapes. And the results of fractal analyses are less accurate (in sense of capturing objects as much realistically as it is possible). Furthermore, formula (1) implies that the longer perimeter of the shape, the higher fractal dimension as a result. And this is very important fact, when calculation using formula (1) is used. Fractal analysis results are then influenced by the factors from logic sequence:

reference scale of map/dataset – generalization degree – perimeter of an area – fractal dimension value

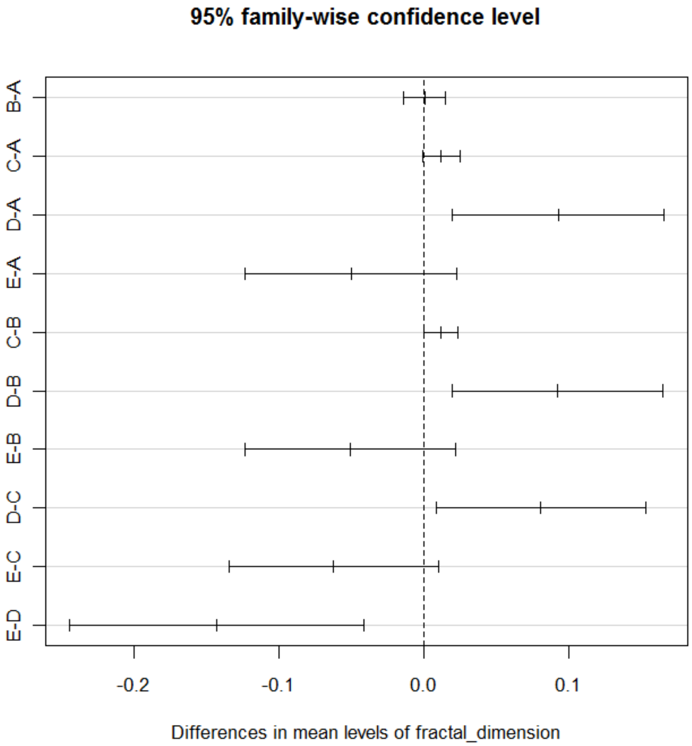


Fig. 4. Graphical output of TukeyHSD. Dotted vertical line is the mean, horizontal lines describe comparison between two classes (meanings of the characters on y-axis are in Table 1).

5 Conclusion

The use of fractal geometry in evaluating land-cover areas was presented. Resulting values of fractal dimension of such areas were commented using expert knowledge of the Olomouc region. Geographical context was mentioned too and proper visualization was made as well. Overall fractal dimension was calculated for comprehension amongst land-cover classes. Finally, some important aspects of generalization influence and CORINE classification system on the results were mentioned.

The paper brings to the reader basics of fractal geometry and its possible usage in geospatial analyses. Brief historical facts are also presented and plenty of publications and papers noted. It is obligatory to introduce methodological frame of fractal geometry apparatus, including formula by which the fractal dimension was calculated. The original case study was carried out to demonstrate practical use of fractal geometry and consequence analyses. As mentioned above, fractal analysis built its stable position in various natural sciences, including geoinformatics and geocomputation

Fractal analyses are very sufficient for measuring complexity or irregularity of various objects, but there are other metric characteristics of the shape (e.g. compactness, convexity, roundness, elongation and others) to evaluate objects, areas in this case, respectively. But the main difference between fractal geometry and this group of metric characteristics is in use of mathematical apparatus and, what is even more important, in concept of fractal geometry and chaos theory. And that is why the fractal geometry built its position in all kind of geospatial analyses.

Acknowledgments. This work was supported by the student project Research of person movement at the intersection of urban and sub-urban area in Olomouc region of the Palacký University (Integral Grant Agency, project no. PrF_2010_14).

References

[1] Batty, M. and Longley, P. (1994): Fractal Cities: A Geometry of Form and Function, Academic Press Ltd., London, San Diego, 1994, 394 p.

[2] De Keersmaecker, M.-L., Frankhauser, P., Thomas I.: Using fractal dimension for characterizing intra-urban diversity: The example of Brussels. Paper presented at the ERSA 2003 Congres, Jyvaskyla, Finland, 27-30 September, 2003.

[3] Falconer, K.J.: Fractal geometry: Mathematical foundations and applications. Chichester, John Wiley & Sons. 1999.

[4] Falconer, K.J.: The Geometry of Fractal Sets. Cambridge University Press, Cambridge, 1985.

[5] Frankhauser, P.: The Fractal Approach. A New Tool for the Spatial Analysis of Urban Agglomerations, Population: An English Selection, Vol. 10, No. 1, New Methodological Approaches in the Social Sciences (1998), pp. 205-240.

[6] Ge, M., Lin, Q.: Realizing the Box-counting Method for Calculating Fractal Dimension of Urban Form Based on Remote Sensing Image. Geo-spatial Information Science, Volume 12, Issue 4 (1 December 2009), pp 265-270. doi: 10.1007/s11806-009-0096-1 December 2009.

[7] Goodchild, M.F.: Fractals and the accuracy of geographical measures. Math. Geol., Vol 12 (1980) , pp 85–98.

[8] Hastings, H. M., Sugihara, G.. Fractals: A User’s Guide for the Natural Sciences. Oxford : Oxford University Press, 1994. 235 p.

[9] Iverson, L. R. : Land-use changes in Illinois, USA: The influence of landscape attributes on current and historic land use, Landscape Ecology vol. 2 no. 1 pp 45-61 (1988), SPB Academic Publishing, The Hague.

[10] Jenerette G. D., Wu, J.: Analysis and simulation of land-use change in the central Arizona - Phoenix region, USA, Landscape ecology, ISSN 0921-2973 , 2001, vol. 16, no7, pp. 611-626 (1 p.1/4), © 2001 Kluwer Academic Publishers. Dordrecht.

[11] Kitchin, R., Thrift, N.: International Encyclopedia of Human Geography. United Kingdom : Elsevier Science, 2009. 8250 p. (hardcover).

[12] Laidre, K.L. et al.: Fractal analysis of narwhal space use patterns. Zoology, 107 (2004), pp. 3–11.

[13] Mandelbrot, B. B. (1967): How long is the coast of Britain? Statistical self-similarity and fractional dimension, Science 155 (1967), pp. 636-638.

[14] Mandelbrot, B. B. (1989): Fractal geometry: What is it and what does it do? In: Fleischmann, M., Tildesey, D. J. a Ball, R. C., 1989, pp. 3-16.

[15] Mandelbrot, B.B.: The Fractal Geometry of Nature. W. H. Freeman. San Francisco, 1983. 458 p.

[16] McAdams, M.A.: Applying GIS and fractal analysis to the study of the urban morpholgy in Istanbul, GEOMED 2007, held on 5-8 June 2007 in Kemer, Antalya, Turkey.

[17] O'Neill, R.V. et al.: Indices of landscape pattern, Landscape Ecology vol. 1 no. 3 pp 153-162 (1988), SPB Academic Publishing, The Hague.

[18] Peitgen, H.-O., Jürgens, H., Saupe, D.: Chaos and Fractals : New Frontiers of Science. New York : Springer, 1992. 984 p.

[19] Peitgen, H.-O., Jürgens, H., Saupe, D.: Fractals for the Classroom : Introduction to Fractals and Chaos. New York : Springer, 1993. 452 p.

[20] Reynoso, C.: The impact of chaos and complexity theories on spatial analysis - problems and perspectives. 24th Research Symposium: Reading Historical Spatial Information from around the World: Studies of Culture and Civilization Based on GIS Data, Kyoto Japan, 7-11 February, 2005.

[21] Saville, D.J.: Multiple comparison procedures: the practical solution. Am. Statistn, 44 (1990), pp. 174-180.

[22] Seto, K. C., Fragkias, M.: Quantifying spatiotemporal patterns of urban land-use change in four cities of China with time series landscape metrics, Landscape Ecology (2005) 20: 871–888, Springer 2005.

[23] Stanley, H.E., Ostrosky, N.: On Growth and Form: Fractal and Non-Fractal Patterns in Physics. Nijhoff, Boston, Mass., 1986.

[24] Tannier, C., Pumain, D.: Fractals in urban geography: a theoretical outline and an empirical example, Cybergeo: European Journal of Geography, document 307, Paris, 2005, 24 p.

[25] Theiler, J.: Estimating fractal dimension. J. Opt. Soc. Am. A, Vol.7, No. 6 (June 1990), pp. 1055-1071

[26] Torrens, P. M., Alberti, M.: Measuring Sprawl, Centre for Advanced Spatial Analysis, Paper 27, London, 2000, 43 p.

[27] Turner, M. G. : Spatial and temporal analysis of landscape patterns, Landscape Ecology vol. 4 no. I pp 21-30 (1990) SPB Academic Publishing, The Hague.

[28] White, R., Engelen, G., Uljee, I. , Lavalle, C. and Ehrlich, D. 2000. Developing an Urban Land Use Simulator for European Cities. In K. Fullerton (ed.), Proceedings of the 5th EC GIS Workshop: GIS of Tomorrow, European Commission Joint Research Centre: 179-190.

Determining Ecotones by Decision Support Systems

Vilém Pechanec, Jan Brus, and Jan Caha

Department of Geoinformatics, Faculty of Science, Palacký University in Olomouc
Tř. Svobody 26, 771 46 Olomouc, Czech Republic
vilem.pechanec@upol.cz, jan.brus@upol.cz, jan.caha@klikni.cz

Abstract The investigation into ecotones enabled a better understanding of the causal relationships between certain landscape elements, landscape utilisation categories and ecotones. By studying ecotones, we wanted to expand understanding of patterns having an influence on the landscape condition, structure, functions, landscape elements and their relationships. The fuzzy approach was applied for better understanding and modelling of ecotones. This methodology was afterwards built in expert system. The landscape assessment process based upon an expert system allows for a multidisciplinary view of a landscape. The landscape is evaluated from several perspectives: the ecological stability, soil erosion, retention capacity and economic landscape calculations by designed expert system. These results from expert system were taken into consideration during mapping ecotones. Ecotones may serve as one of the distinctive indicators of the impact humans have on the landscape. It is therefore necessary to develop more sophisticated methods of their studying.

1 Introduction

Ecotones are significant regions of landscape heterogeneity that contain elements, patterns, and processes existing and operating at varying spatial scales [19], [14], [24]. Ecotones play a pivotal role in landscape. This can be seen from several viewpoints — environmental, biological, economic, historic, aesthetic, etc. Their most crucial task is ecological. They represent specific ecosystems in a landscape, corridors for the migration of animals or distribution of plants. They also contribute to soil protection against erosion, etc. During their existence ecotones go through developmental stages which depend primarily on the dynamics of factors in the surrounding environments. Some ecotones are temporally stable some may migrate or mutate. Spatial relationships within the surrounding environments are of great importance since ecotones are the zones where communities, populations of plants and animals meet, where they compete, create some tension and blend. In nature ecotones often function as corridors or barriers enabling or restricting the flow of mass or energy. The temporal and spatial stability of ecotones contributes to the ecological value of the ecotone community and to the greater value of a landscape.

V. Snášel, J. Pokorný, K. Richta (Eds.): Dateso 2011, pp. 206–215, ISBN 978-80-248-2391-1.

We can classify ecotones by various aspects, e.g. by structure. It is difficult to perfectly simulate the real world, due to its complexity. It has been argued that uncertainty information is a vital component in the use of spatial data for decision support [11], [1]. We are forced to model uncertainties which suggest fuzzy logic to problem solving. Many techniques have been developed for communicating uncertainty in data and models for specific visualization applications, such as remote sensing, land allocation, [3], [16], [1]. Development in the field of Geographical Information Systems (GIS) facilitates the integrating of flexible decision support functionalities to perform multicriteria evaluations to solve allocation or location problems, to perform suitability analysis, to integrate different criteria for options choice and group decisions [13], [18]. For such a specific and complex field as ecotones there is compulsory to use unique technique. Fuzzy approach can be applied to help to better mapping and modelling, due to non-sharp nature of ecotones, also their uncertainties. Fuzzy theory is practically implemented through decision-making processes in expert system. Build in expert knowledge helps to coping with vague concepts such as imprecise and uncertain values of attributes of spatial entities of ecotones. This research was mainly focused on testing possibilities of using Spatial Decision Support Systems (SDSS) for mapping ecotones.

2 Theoretical background

2.1 Fuzzy theory

Sometimes crisp information isn’t the best approximation of reality. This is the main reason for use of fuzzy set theory and fuzzy logic in GIS and decision making process. This point of view is shared by many authors [6], [8], [12], [22] and [23]. The best overview of use of fuzzy sets in modelling geographical elements and phenomena are presented in [6] and [8]. Concepts of fuzzy set theory and fuzzy logic were firstly presented by L. A. Zadeh in 1968 [25] in his article "Fuzzy sets". Compared to classical set theory where elements are evaluated like belonging or not belonging to the set, fuzzy set theory uses membership function to assess each element's membership value. Membership value is the number from interval [0, 1] where 0 means that element is not included in set. Value 1 means that element is fully included in the set, any other value from given interval means that element is a fuzzy member of the given set. The higher of the membership value predicate how much the element belongs to this set Fig.1. Within fuzzy set theory, vague and imprecise numeric values can be represented by fuzzy subsets on the basic numeric domains. There are several ways how can we define membership values for elements, for finite sets we can clearly define membership value for each element and for infinite sets (i.e. real numbers) we can define a function, named membership function, which defines membership value for each element of the set.

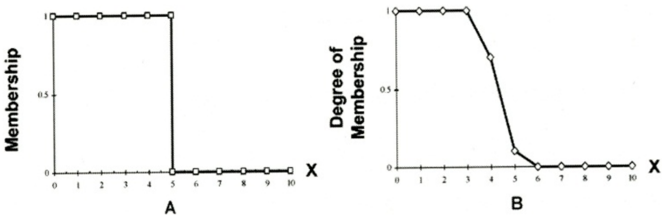


Fig. 1 Boolean versus Fuzzy sets [11]

Unlike from the crisp set theory where element either belongs to set A or B with fuzzy sets it is possible for an element to belong to both sets A, B to each with different membership value. Fuzzy logic is derived from those assumptions. Fuzzy logic is a multi-valued logic that deals with reasoning where classic binary sets are not appropriate.

3 Technical background

3.1 SDSS

In recent years, the GIS is increasingly understood as a means used to support decision-making and recognized as the basis for the Spatial Decision Support Systems (SDSS). SDSS are a specific kind of information system. There is no unambiguous and generally accepted definition because forms of technology have not been profiled yet. However, the majority of authors agree that it is a spatial expansion of Decision Support Systems (DSS), or rather the integration of GIS and DSS. SDDS are computer information systems that provide support for problems difficult to formulate and structure. They are usually considered when it is impossible to use a fully automated system. SDSS are closely related to knowledge-based and expert systems whose creation was possible due to artificial intelligence. SDSS also provide detailed displays resulting in reduced decision time and enabling a better grasp of spatial problems due to better visualization of the problem to be solved [16].

In relation to the previous paragraph, we can set apart special category of SDSS which are expert systems. They can stand alone, but in combination with GIS they integrate into a very powerful tool. We must emphasize here that the possibility of easy handling without deeper knowledge of processes and algorithms may lead to a completely incorrect interpretation of results, and thus, erroneous decisions.

3.2 Expert systems

Expert systems are computer programs able to simulate actions of an expert in a particular field when solving complicated tasks . Other authors [**20**, **5**] describe expert system as software or combination of software and hardware, which can complete the exercise of specific complex tasks. These tasks can also be solved by a human expert, but require significant expertise in the solution. Expert systems provide a powerful tool for solving many problems that often cannot be solved by other, more traditional methods. The usage of expert system has proved to be crucial in the process of decision support and problem solving [8]; therefore, their usage has spread into many sectors. They are considered a sub-category of knowledge-bases systems. They are based on symbolic representation of knowledge and its implementation in an inference mechanism. Experts in the given field present the source of knowledge and procedures. These systems are able to justify solution procedures. They are used primarily for tasks difficult to structure and algorithmize, e.g. problems with recognition of situations, diagnosis of status, construction, planning, monitoring of status, corrections, management and decision-making. However, experience and intuition have to be part of the solution. Certain authors [10] look at the expert analyst required to operate the system as posing a barrier to decision makers who must translate the problem into a form that can be understood by experts who, in turn must translate their understanding of the problem into a form that can be evaluated and solved [21].

3.3 Ecosystem Management Decision Support (EMDS)

EMDS - is a product which if interconnected with ArcGIS provides a comprehensive SDSS product. The supplement comes from the Pacific Northwest Research Station U.S. Forests Service. According to its authors it integrates logical formalism justified on the basis of the knowledge base in the GIS environment. It provides support for decisions on evaluation and assessment of landscape from an ecological point of view. The EMDS decision-making pattern is based on a knowledge base that uses fuzzy logic, network architecture and object-based approach. The basic architecture of objects of EMDS knowledge bases enables an increase in development of dependent complicated knowledge data. Up-to-date modern research methods dispose of mathematical models characterised by very specific mathematical dependencies between the status of monitored objects and the processes influencing them. Fuzzy logic tools significantly enhance the ability to work with incomplete or vague information. The proposed network architecture of EMDS knowledge bases allows evaluation of the influence of the missing information and has the ability to come to conclusions with incomplete information. The current version of EMDS is the 4.1 version that ensures compatibility with ArcGIS 9.x and includes a newly designed hotlink browser tool, which speeds up work and provides graphical representation of the knowledge base for landscape features chosen from topics intended for analysis. Internet presentation of maps is secured via ArcIMS.

4 Methodology

Many researches were done in the field of determining ecotones [9, 15]. Comprehensive study based on representing transitional zones and determining their borders was also carried out [2]. Based on upper text, fuzzy theory brings apparatuses how to deal with specific fuzzy sets. In the case, that we accepted the representation of ecotones by fuzzy sets, whole ecotone same as its transitional borders can be modelled.

The research on ecotones was carried out in the catchment area of the stream Trkmanka in South Moravia. The boundaries of the region are formed by the boundaries of the Trkmanka catchment area as taken from the hydrological map. The Trkmanka catchment area comprises the regions of Břeclav, Hodonín, and Vyškov in the south-east Moravia. This narrow territory drops from the north-east to the south-west. Its area covers approximately 380 km2. The Trkmanka catchment area lies in the Carpathian part of the Czech Republic. It consists of the flysh belt of the outer part of the Western Carpathians and the Vienna Basin. Its prevailing parts are formed by sedimentary fill. Lowest parts have a flat alluvial relief and belong to the vale Dolnomoravský úval. Three modelling areas were Kobylí, Ždánice and Rakvice.

The landscape assessment process based upon an expert system allows a multidisciplinary view of a landscape. The landscape was assessed from four viewpoints according to the selected indicators. Monitored ecological indicators mainly include an ecological stability coefficient (1st indicator) showing an ecosystem’s ability to compensate for changes caused by external factors in order to keep its natural properties and functions. This closely relates to the erosion hazard (2nd indicator) and landscape storage capacity (3rd indicator). A modern tool for nature conservation is the economic assessment of ecosystems (4th indicator) and their non-productive functions. It enables us to compare ecological values and economic profits in the same terms and hence provides for better reasoning in decision-making processes. In the assessment the ecological value of nature is always taken into consideration.

Whole process of evaluating landscape stands as primary phase in determining of ecotones. Results computed by expert system are primary data which were used as initial data in the ecotones determined by botanist. Such an assessment can be performed with common accessible data, including the land use, biotype mapping of the Czech Republic which was processed by methodology introduced by NATURA 2000, pedoecological unit (soil-ecological unit in Czech terminology, used for land appraisal), forest topology and contour lines. A layer of “soil ecological units for soil rating” was nationally special. Soil ecological unit for soil rating of the agricultural parcels is a five-digit numerical code of the main soil and climatic conditions that affect the productive capacity of agricultural land and its economic valuation. Forest typology is the classification system consisting of differentiation in the management of the forest lands. This is a nationwide database of permanent environmental conditions. This database standardizes the potential natural vegetation in relatively homogeneous territorial units in forests. It is necessary for economic planning of forest management. Described layers are finally set up into specified structure and evaluated in system Assessment in Ecosystem Management Decision Support (EMDS). In many real cases, the available data are crisp, precise, but they hold uncertain on their reliability for several reasons: either because the agencies that are the source of the data cannot be entirely trusted, or because one knows that the means of acquisition are not enough sophisticated and generate systematic errors; not least because data are a result of a subjective analysis, such as surveyed data. Uncertainty on precise or imprecise data can be represented by associating a degree of confidence or credibility, or reliability with them [4].

5 Results

Three methodologies based were used to determine the indicator of landscape stability coefficient. The data was valid for the year 2007 and the analysis did not cover the whole study area, but only parts. We obtained quite varying values for particular land categories from the category of above-average land use with distinct disturbances in the natural structures to the category of natural landscape or landscape close to it. Another indicator of landscape stability is long-term soil loss. It was determined using a Universal Soil Loss Equation (USLE) to assess the danger of water erosion to agricultural land (Fig. 4). It gives the potential amount of soil which could be removed due to water sheet erosion. It however does not include its deposit at the site or into the lower layers underneath. This parameter is directly linked to direct runoff. The Runoff Curve Number Method (CN) was applied to determine the value (Fig. 3). All model areas have the greatest representation of values for the zero runoff. The greatest runoff values were those for urban areas and roads. In addition, the modified Hessen biotope assessment method was applied to that part of the study area if the required data was available. The price of one segment means the average costs of increasing the land value by one ecological degree for one square meter of land. In the view of mathematics, ecotones are areas where the particular points can, with a certain probability, be considered as belonging to one ecological stratum or, with another probability, to another one.

Assessment of landscape by expert system based on EMDS system is composed of several parts. Mainly from algorithmized decision schema for appraisal landscape segments (network in NetWeaver for EMDS), simultaneously is formed and filled knowledge base about landscape. NetWeaver knowledge bases use an object-based approach, which makes them very modular; therefore, they are easily created and maintained. Moreover, the system enables interactive tuning in an arbitrary stage of the creation of the knowledge base. This significantly speeds up the development process. Fuzzy logic provides calculation methods which do not require directive expression. NetWeaver is completely object-oriented system. This means that networks and data links are programming objects that represent or substitute objects or notions of the real world (Reynolds, 1998). Implementation of this knowledge embodied in interconnection between an appraisal network and input entry about landscape with datalinks in network. Optimized data model with series of reclassification tables has to be done for connection with entry data. Assessment of landscape segments by an appraisal network is achieved finally in EMDS.

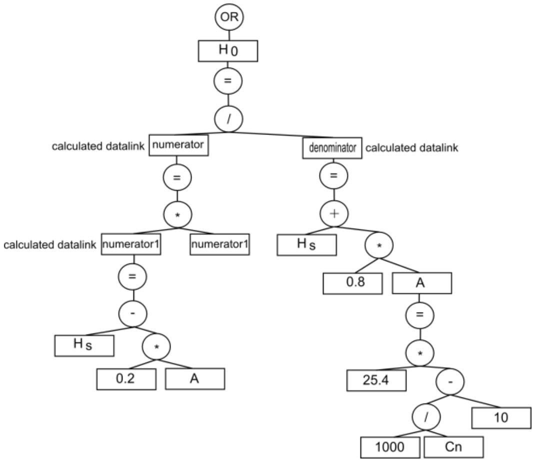


Fig. 3. An appraisal grid for the Runoff Curve Number Method (CN method)

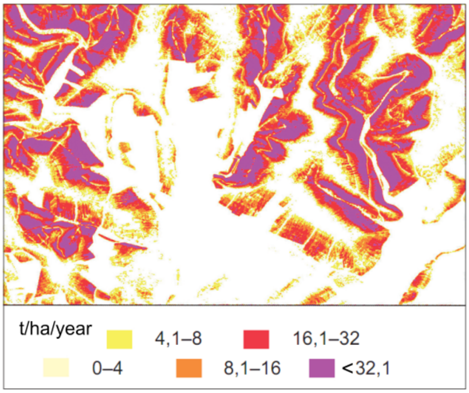


Fig. 4. The long-term average soil loss of agricultural land and forestland in a model area of Ždánice

The highest ecological stability is calculated in modeled area Ždánice. The lowest endangered area by water erosion calculated by USLE (Universal Soil Loss Equation) is in modeling area of Kobylí. At suggested precipitation 12 mm is a level of straight runoff equated to 0. Appraisal of landscape by the modified Hessen method is also significant due to the evaluation of landscape. All this information is taken into consideration for the final evaluation of possible occurrence of ecotones. Predicted areas were faced with data of field survey and aerial photography.

6 Conclusion

The main idea of this work was practically to test possibilities of SDSS capability in GIS software for mapping remarkable landscape structure. Role of ecotones in the landscape is important and ecotones are formed in a special part of the landscape. They can be distinguished mainly by field work which was proofed by botanists, because of their blur border and many ecological factors which affect them. It was extremely difficult to model all conditions, which come into the process of developing ecotones. Only datasets which build up the expert system were not efficient to map ecotones in the study area. In the other hand, in our model was not decided to calculate with this amount of factors. It was mainly considered as a pilot and testing application for help with mapping ecotones. As the one of the crucial problems were considered uncertainties in datasets and also mapping measure of GIS analysis. This measure was derived from the measure of less detailed map layer. Uncertainty of entering data is mainly based on the form how the agencies collect them. This data have to be analysed and tested for accuracy in the field, due to their error. The second problem was based on problematic expression of ecotones in fuzzy theory in GIS. Basic ideas of use of fuzzy set theory and fuzzy logic in geography and geoinformatics can be found in many sources including [6], [8], [12] and [17]. Almost no of those sources provide some kind of workflow how to deal with imprecise data in GIS. As suggested in [6] that this fact may be caused by lack of tools for work with fuzzy sets and fuzzy logic in the widespread GIS. The problem probably lays deeply in the structure of current GIS. As most of today’s software GIS highly rely on mathematics and informatics which structure is extremely connected to both Boolean logic and crisp set theory. Cause of this is probably fact that alternative theory to crisp set theory and Boolean logic exists only since 1960’s but both other are much older and thus more rooted in mathematics and informatics. EMDS was chosen as final tool for developing a appraisal grid. EMDS including NetVeawer was considered as a very powerful tool with large field of application. Ability of implementation of fuzzy sets into expert systems is in this case inestimable. Final results of this research stand only like partially data, which can be used as base point for future analysis of ecotones in the landscape. Future visions stand on developing more sophisticated expert system, which will better utilize fuzzy theory and will implemented more variables into the appraisal grid for assessment the landscape. This approach of mapping by expert system will be subsequently applied in a recent project such as monitoring sub-urban areas.

Acknowledgments

This work was supported by the Czech Science Foundation grant GA205/09/1159. The intelligent system for interactive support of thematic map design.

References

1. Aerts, J. C. J. H., K. C. Clarke, et al. (2003) Research Papers - Testing Popular Visualization Techniques for Representing Model Uncertainty. Cartography and geographic information science. 30: 249 (14 pages). 5
2. Arnot, C., Fischer, P. (2007) Mapping the Ecotone with Fuzzy Sets. In Morris, A., Kokhan, S. (eds.) (2007): Geographic Uncertainty in Environmental Security, p. 19-32.
3. Aspinall, R.J., Pearson D.M. (1995) Describing and managing uncertainty of categorical maps in GIS. In: Fisher P (ed) Innovations in GIS 2. Taylor & Francis, London, pp 71–83
4. Bordogna, G. et al. (2007) A flexible decision support approach to model ill-defined knowledge in GIS. In Morris, A., Kokhan, S. (eds.) (2007): Geographic Uncertainty in Environmental Security, p. 19-32.
5. Boss, R. W. (1991) “What Is an Expert System?” ERIC Digest. ERIC Clearing House on Information Resources, Syracuse, NY, 1-3.
6. Burrough, P., McDonnell, A., Rachael A. (1998) Principles of Geographical Information Systems. Second Edition: Oxford University Press, 1998. 356 p. ISBN 0198233655.
7. Crossland, M. D., Wynne, B. E. and Perkins, W. C. (1995) Spatial Decision Support Systems: An overview of technology and a test of efficacy, Decision Support Systems, 14, 3, 219-235.
8. Dragićevic, S. (2005) Multi-Dimensional Interpolations with Fuzzy Sets. In: Petry, F.E.; Robinson, V.B.; Cobb, M.A. (eds.) Fuzzy Modeling with Spatial Information for Geographic Problems. Berlin: Springer, ISBN 3-540-23713-5.
9. Fortin M. J., Olson R. J., Ferson S., Iverson L., Hunsaker C., Edwards G., Levice D., Butera K. and Klemas V. (2000) Issues related to the detection of boundaries. Landscape Ecology 15: 453–466.
10. Goodchild, M. F., (1992) Geographical information science. International Journal of Geographical Information Systems, 6(1), 31–47.
11. Hunter, G.J., and Goodchild, M.F. (1995) A new model for handling vector data uncertainty in geographic information systems. Proceedings, URISA, San Antonio, Texas, pp. 410-419.
12. Hwang, S., Thil, J.C. (2005) Modelling Localities with Fuzzy Sets and GIS In: Petry, F.E.; Robinson, V.B.; Cobb, M.A. (eds.) Fuzzy Modeling with Spatial Information for Geographic Problems. Berlin: Springer, ISBN 3-540-23713-5.
13. Jankowski, P., Nyerges, T. (2001) Geographic Information Systems for Group Decision Making, Taylor & Francis: London. March, 2001. equal contribution; eight chapters synthesizing research 1995-2000.
14. Ji, M. (2002) Fuzzy modelling of African ecoregions and ecotones using AVHRR NDVI temporal imagery. Geocarto International 17(1): 21-30.

15. Kent, M., Gill, W. J., Weaver, R. E., and Armitage, R. P. (1997) Landscape and plant community boundaries in biogeography, Progress in Physical Geography 21: 315-353.
16. Leitner M, Buttenfield BP (2000) Guidelines for display of attribute certainty. Cartography and Geographic Information Sciences 27:3–14.
17. Li, Z., Zhou, Q., Kainz, W. (eds.) (2004) Advances in Spatial Analysis and Decision Making: Proceedings of the ISPRS Workshop on Spatial Analysis and Decision Making. Lisse: Swets & Zeitlinger, ISBN 90-5809-652-1.
18. Malczewski, J. (1999) GIS and Multicriteria Decision Analysis, John Wiley and Sons, 392 pp., New York, NY.
19. Nellis, M.D., Briggs, J.M. (1989) The effect of spatial scale on Konza landscape classification using textural analysis. Landscape Ecology 2(2): 93-100.
20. O'Looney, J. (2000) Beyond maps: GIS and decision making in local government. Redlands, Calif: ESRI Press.
21. Popper, M., Keleman, J. (1998) Expertné systémy. Bratislava. Alfa.
22. Verstraete, J., Hallez, A. and De Tre, G. (2007) Fuzzy regions: theory and applications. Geographic Uncertainty in Environmental Security, pp. 1-17. Springer, Dordrecht
23. Verstraete, J., Hallez, A. and De Tre, G. (2007) Fuzzy regions: theory and applications In Morris, A., Kokhan, S. (eds.) (2007) Geographic Uncertainty in Environmental Security, p. 19-32.
24. White, J.D., Running, S.W., Ryan, K.C. and Key. C.C. (2002) Fuzzy logic merger of spectral and ecological information for improved montane forest mapping. Geocarto International 17(1): 59-66.
25. Zadeh, L.A. (1968) ‘Probability Measures of Fuzzy Events’, Journal of Mathematical Analysis and Applications, 23, 421-427.

Orthophoto Map Feature Extraction Based on Neural Networks

Zdeněk Horák[1], Miloš Kudělka[1], Václav Snášel[1], and Vít Voženílek[2]

[1] Department of Computer Science, FEI, VSB - Technical University of Ostrava,
17. listopadu 15, 708 33, Ostrava-Poruba, Czech Republic
{zdenek.horak, milos.kudelka, vaclav.snasel}@vsb.cz
[2] Department of Geography, Faculty of Science, Palacky University, Olomouc
tr. Svobody 26, 771 46 Olomouc, Czech Republic
vit.vozenilek@upol.cz

Abstract. In our paper we use neural networks for tuning of image feature extraction algorithms and for the analysis of orthophoto maps. In our approach we split an aerial photo into a regular grid of segments and for each segment we detect a set of features. These features describe the segment from the viewpoint of general image analysis (color, tint, etc.) as well as from the viewpoint of the shapes in the segment. We also present our computer system that support the process of the validation of extracted features using a neural network. Despite the fact that in our approach we use only general properties of an images, the results of our experiments demonstrate the usefulness of our approach.

Keywords: orthophoto map, image analysis, neural network

1 Introduction

Aerial data is one of the standard sources for the extraction of topographic objects. Classical applications include the detection and extraction of roads and buildings. It may also include other objects such as forests and vegetation, agricultural use and parcel boundaries, hydrography, etc. Currently, this field of analysis falls under the paradigm of Object-based Image Analysis (OBIA), which is a sub-discipline of geoinformation science devoted to partitioning remote sensing imagery into meaningful image-objects with a focus on the generation, modelling and classification of these objects (see [4]).

Existing methods can be divide into two groups - automatic and semi-automatic. Semi-automatic methods – as opposed to the automatic ones – require human intervention, especially when tuning algorithms and judging results. Because of the many influences that contribute to the quality of aerial imagery, we usually cannot fully rely on automatic methods.

In our approach we proceed in the same way. For a description of images we use a set of general features (we do not use any knowledge base of known objects for extraction). These features are detected by a set of specific algorithms. We use a neural network for tuning these algorithms. The features are then detected automatically. To assess the quality of the detection we use our own application that incorporates the neural network again. This application visualizes how the system assesses a particular images. In the case of discovered inaccuracies, the user can retroactively affect the parameters of automatic feature detection.

In the following section we discuss related approaches. The third section contains a description of features in our detection system, while the fourth section recalls some basics of tools and techniques used. The fifth section is focused on our experiment with orthophoto maps.

2 Related approaches

The basis for all methods and algorithms for analyzing the orthophoto maps is digital image processing. Digital image processing is a set of technological approaches using computer algorithms to perform image processing on digital images ([11], [16], [14]). Digital image processing has many advantages over analogue image processing. It allows a much wider range of algorithms to be applied to the input data and can avoid problems such as the build-up of noise and signal distortion during processing ([7]). Digital image processing may be modeled in the form of multidimensional systems rather than images that are defined over two dimensions (perhaps more). Some research deals with a new object-oriented classification method that integrates raster analysis and vector analysis (e.g. [10]). They combine the advantages of digital image processing (efficient improved CSC segmentation), geographical information systems (vector-based feature selection), and data mining (intelligent SVM classification) to interpret images from pixels to objects and thematic information.

Many different approaches dealing with the detection and extraction of man-made objects can be found in [2]. These are mainly methods focused on automatic road extraction and automatic building extraction. A summary and evaluation of methods and approaches from the field of automatic road extraction can be found in [13], while for the field of building extraction see [12]. For more recent approaches from the field of Object-Based Image Analysis (OBIA) you can see e.g. [4], a detailed summary of existing methods is described in [3].

Authors of this paper have participated in the development of a commercial Document Management System, which is used in several institutions of the Government of Czech Republic. Experiments based on image dataset from one of these institutions are described in [8].

3 Image features

Our approach is to describe any image in terms of image contents and in the concepts which are familiar to the users. In the following we present features we are capable of detecting. Some of the features are related to the whole image only, but many of them can also be used to describe some parts of the image.

3.1 Color features

According to used colors we are able to find out whether the image is gray-scaled, and if not, whether the image is toned into some specific hue. Also we can say if the image is light, dark or if the image is cool or warm.

- grey-scaled images
- color-toned images
- bright or dark images
- images with cool or warm color tones

The last group of features is color features. We want to describe the image in terms of colors in the same way as a human will, but it does not suffice only to count the ratio of one color in the image or in some area of the image. A more complex histogram is also not enough. We should consider things like dithering, JPEG artifacts and the subjective perception of colors by people. Using color spatial distribution, color histograms and below mentioned shapes recognition we are also able to detect the background color. The colors we are currently able to detect are:

- red, green, blue, yellow, turquoise, violet, orange, pink, brown, beige, black, white and gray
- background color

Color features detection Low-level color features were detected using a combination of their spatial distribution and a comparison with their prototypes. The first version of the system contained prototypes that were constructed manually using our subjective perception. However this approach was not general enough, therefore we have created a set of training image patterns with manually annotated color features. To deal with human perception we have averaged the annotation results among several annotators. Using this set we have trained the artificial single-layer feed-forward neural network (see [17], [1]) to confidently identify the mentioned features. This network was very similar to the network used in the whole application, which is described below in detail.

As an input we have used the pixels of particular patterns in different color models (as different models are suitable for different color features). Trained neurons (their input weights and hidden threshold) were then transformed (using the most successfull color model) into the color feature prototypes (see fig. 1). We detect all of the mentioned features as fuzzy degrees, but for selected applications we scale them down to the binary case.

Image segmentation To obtain more precise information about the processed image, we have decided to employee an image segmentation technique. Using the Flood fill algorithm (with eight directions, for details see [6]) we were able to separate regions with same (or almost same) color. But to be able to index these shapes, we need to describe them. We have calculated the center of this shape and using this point and different angles we have sliced the shape into several

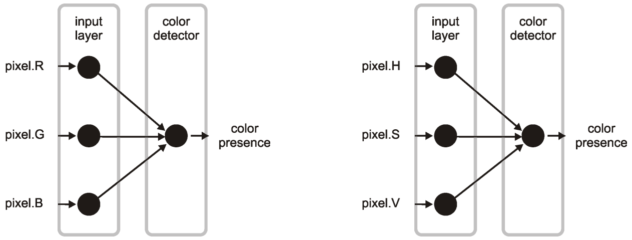


Fig. 1. Illustration of simple neural network color detector

regions (see 16 regions in figures 3, 4, 5). For each region, we have computed the maximum distance from the center. Following the changes of this distance (peaks, regularity) we are able to distinguish between different basic shapes (rectangle, circle, triangle, etc.).

Of course, this approach is not general. We use it only for bigger shapes and we ignore possible holes within the shapes. Because we use mostly downsampled versions of source images, we can guarantee the effectivness of processing. And because we use high quality downsampling, our results are similar to a person's first glance. The shapes we are able to detect are: line, restangle, circle, triangle and quad.

At this moment, we detect shapes separately, but to the resulting description we save only information, whether at least one shape of such kind has been detected (i.e. the image contains one triangle) or whether there are multiple shapes of such kind (i.e. the image contains more triangles).

Currently we are thinking of using obtained distances not only for shape identification, but also for shape description. The same shape can be scaled, moved or rotated on different images, but the description using relative distance changes is still the same (up to index rotation). The more different angles we use, the more precise description we obtain.

Anomalies Since the orthophoto maps are created from long distance, interesting objects are often relatively small and vaguely bound in the image. For this reason we have incorporated the concept of anomalies (see [5] for a recent survey). As an anomaly we consider:

- a shape formed by similar pixels,
- which – due its size – cannot be reliably classified as being one of the previously mentioned shapes and
- has other than background color.

As you will see in the experiment section, this concept became very important in our approach. Figure 2 contains highlighted samples of various shapes detected in the orthophoto maps.

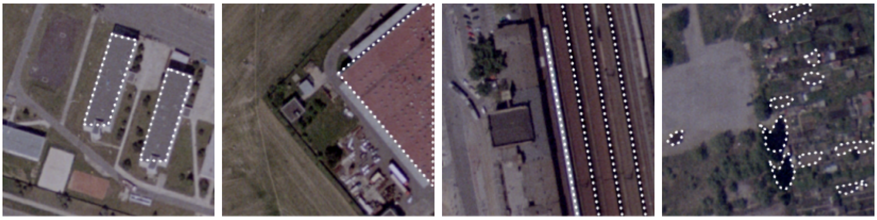


Fig. 2. Various shapes detected in orthophoto maps - rectangles, triangles, lines and anomalies

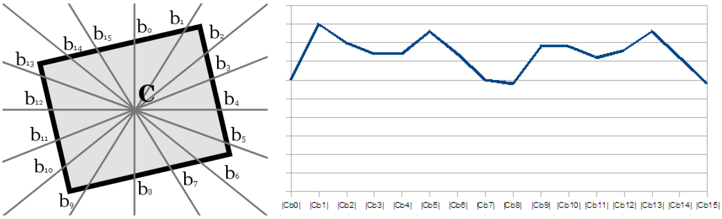

C
b₀
b₁
b₂
b₃
b₄
b₅
b₆
b₇
b₈
b₉
b₁₀
b₁₁
b₁₂
b₁₃
b₁₄
b₁₅
|Cb0| |Cb1| |Cb2| |Cb3| |Cb4| |Cb5| |Cb6| |Cb7| |Cb8| |Cb9| |Cb10| |Cb11| |Cb12| |Cb13| |Cb14| |Cb15|

Fig. 3. Rectangle detection

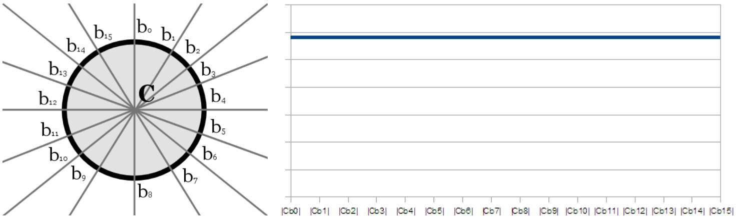

C
b₀
b₁
b₂
b₃
b₄
b₅
b₆
b₇
b₈
b₉
b₁₀
b₁₁
b₁₂
b₁₃
b₁₄
b₁₅
|Cb0| |Cb1| |Cb2| |Cb3| |Cb4| |Cb5| |Cb6| |Cb7| |Cb8| |Cb9| |Cb10| |Cb11| |Cb12| |Cb13| |Cb14| |Cb15|

Fig. 4. Circle detection

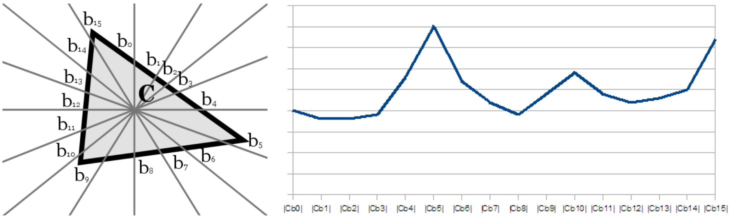

C
b₀
b₁
b₂
b₃
b₄
b₅
b₆
b₇
b₈
b₉
b₁₀
b₁₁
b₁₂
b₁₃
b₁₄
b₁₅
|Cb0| |Cb1| |Cb2| |Cb3| |Cb4| |Cb5| |Cb6| |Cb7| |Cb8| |Cb9| |Cb10| |Cb11| |Cb12| |Cb13| |Cb14| |Cb15|

Fig. 5. Triangle detection

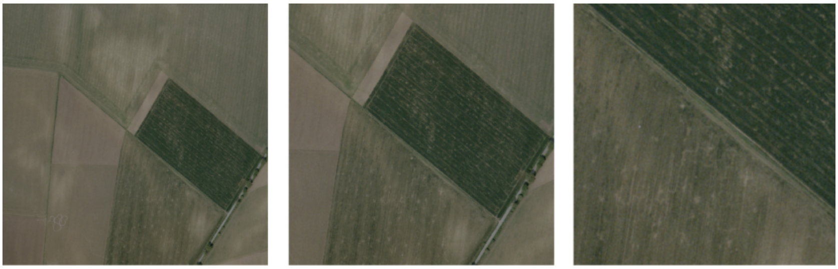


Fig. 6. Ambiguity of shape detection caused by splitting map using different resolutions

4 Preliminary: Neural networks

The **Neural network** (or more precisely artificial neural network) is a computational model inspired by biological processes. This network consists of interconnected artificial neurons which transform excitation of input synapses to output excitation. Most of the neural networks can adapt themselves. There are many different variants of neural networks. Each variant is specific in its structure (whether the neurons are organized in some layers, whether the neurons can be connected to themselves, etc.), learning method (the way the neural network is adapted) and neuron activation function (the way the neuron transforms input excitation to output excitation) and its parameters.

For our purposes we use a structure consisting of an input layer of neurons, several inner hidden neuron layers and one output layer of neurons. As the learning method we use **supervised learning**, where the network is presented repeatedly with specific samples, which are propagated towards the network output. This output is compared with expected results and the network is (using calculated error) adapted to minimize this error. The learning finishes after a predefined number of learning epochs or if the error rate decreases under a predefined constant. After the learning phase, the neural network can be presented with another group of samples and provides its output. For more details on neural networks consult [17], [1] or see [15] for this particular case.

Our particular network is illustrated in figure 8. We have decided to use classical bipolar-sigmoid (because we needed to represent both positive and negative examples) as an **activation function** of neurons (having $\beta = 2$):

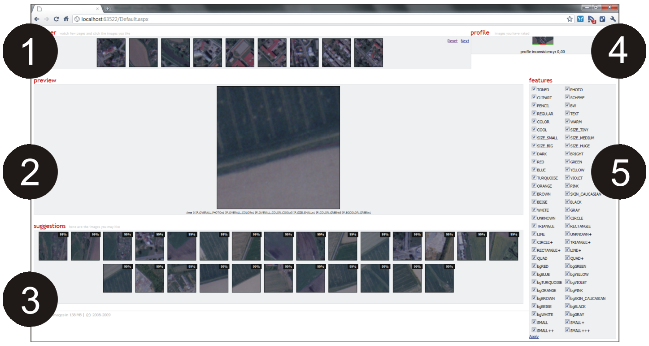
$$f(x) = \frac{2}{1 + e^{-\beta x}} - 1$$

Simple **backpropagation** has been used as a learning algorithm:

$$\triangle w_{ij}(t+1) = \eta \frac{\sigma E}{\sigma w_{ij}} + \alpha \triangle w_{ij}(t)$$

The basic idea of this algorithm is to calculate the total error E of the network (computed by comparing real outputs of the network with expected ones) and then change the weights $\triangle w_{ij}(t+1)$ of the network to minimize this error. The learning rate parameter η controls the speed of weight changes. To speed up learning, we use momentum α – which updates the weight in each step also with the value from the previous step $\triangle w_{ij}(t)$.

5 Feature validation



☑ TONED
☑ CLIPART
☑ PENCIL
☑ REGULAR
☑ COLOR
☑ COOL
☑ SIZE_SMALL
☑ SIZE_BIG
☑ DARK
☑ RED
☑ BLUE
☑ TURQUOISE
☑ ORANGE
☑ BROWN
☑ BEIGE
☑ WHITE
☑ UNKNOWN
☑ TRIANGLE
☑ LINE
☑ CIRCLE+
☑ RECTANGLE+
☑ QUAD
☑ bgRED
☑ bgBLUE
☑ bgTURQUOISE
☑ bgORANGE
☑ bgBROWN
☑ bgBEIGE
☑ bgWHITE
☑ SMALL
☑ SMALL++
☑ PHOTO
☑ SCHEME
☑ BW
☑ TEXT
☑ WARM
☑ SIZE_TINY
☑ SIZE_MEDIUM
☑ SIZE_HUGE
☑ BRIGHT
☑ GREEN
☑ YELLOW
☑ VIOLET
☑ PINK
☑ SKIN_CAUCASIAN
☑ BLACK
☑ GRAY
☑ CIRCLE
☑ RECTANGLE
☑ UNKNOWN+
☑ TRIANGLE+
☑ LINE+
☑ QUAD+
☑ bgGREEN
☑ bgYELLOW
☑ bgVIOLET
☑ bgPINK
☑ bgSKIN_CAUCASIAN
☑ bgBLACK
☑ bgGRAY
☑ SMALL+
☑ SMALL+++

Fig. 7. Screenshot of the validation application with highlighted parts of its UI - (1) image gallery, (2) image preview, (3) suggested images, (4) user profile, (5) considered features

To verify that our set of features is capable of representing the user point of view on the images content, we have created a web application for image suggestion. In the first step, the user is presented with several random images from the data. He/she marks these images as interesting (or not interesting). Using this process the user search profile is created. In the second step the application tries to understand this profile (a set of positive and negative examples) using an artificial multilayer feed-forward neural network. In the last step, the trained network is presented with the whole dataset and suggests images which may be potentially interesting to the user.

The user can clarify his/her profile by marking further images and the process is repeated. We have used part of the profile for training and the rest for the validation of the profile to verify the meaningfulness of this profile. The score of presented images is an indication for users to add more positive (if the overall score is too low) or negative (the overall score being too high) examples.

Network parameters Parameters of the neural network have been selected as follows (see fig. 8): 610 input neurons (input activation represents the degree of individual feature presence), 5 hidden neurons and one output neuron (representing the degree of image acceptance). The learning rate was $\eta = 0.1$ and momentum $\alpha = 0.1$. The maximum number of iterations per learning epoch was set to 1,000. The number of input neurons correspond to the number of features

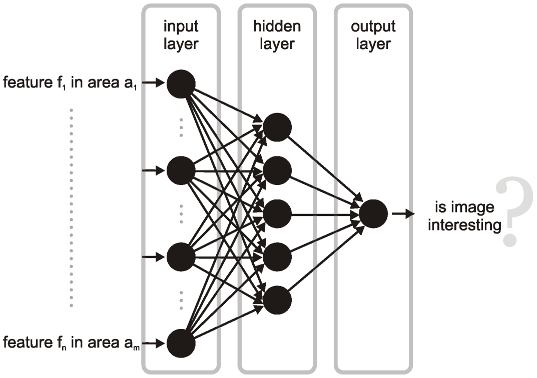


Fig. 8. Illustration of neural network used in application

in different regions of the image. Remaining parameters have been selected after several attempts of being subjectively the best. A larger number of hidden neurons often caused the overtraining of the network (good performance on the training samples with very limited ability of generalization). A larger number of iterations produced no significant improvement. Lower values failed to comply with user judgments.

Application description This application (see fig. 8) has been created as an ASP.NET Web application on the Microsoft .NET platform utilizing several other technologies such as CSS/JavaScript to improve user experience. Most of the computation time is used during the image dataset indexing, which is done only once and can be precomputed offline. The indexing of particular image takes on average 0.76 seconds and can be easily paralelized as the indexing of every particular image is a completely independent.

The neural network is recreated with every request, but in high-load environment can be stored between the requests. Application memory contains indexed image signatures only, therefore the whole application is well scalable. In our testing environment we have been able to run this application easily on an Intel 2.13 GHz processor and 4 GB RAM with a dataset containing several thousand images. Clearly the process of running the neural network with particular image signatures in every step of recommendation has its computational limits, but these limits lie far beyond the boundaries of the purpose of our experiment.

We have performed several user testing sessions where we have selected the presented set of features as being the most suitable for our purposes. Using this process we made sure that normal users can understand image analysis systems based on selected features and these users were normally able to find expected results after giving two or three positive and negative examples. The discrepancy between the user's expectations and the output of the system is a suggestion for another iteration of feature detection tuning.

6 Conclusions and future work

Using several mathematical models and methods, such as neural networks, we have developed and described a system which can analyze orthophoto maps, detect user-oriented features in the maps and visualize the structure of the region. In the future work we will investigate the similarities between different image segments and sources of these similarities. We would like to consider different kinds of maps and use our approach on a much wider landscape region.

Acknowledgment

This work is supported by Grant of Grant Agency of Czech Republic No. 205/09/1079.

References

1. Arbib M.A.: The handbook of brain theory and neural networks, The MIT Press (2003)
2. Baltsavias E.P., Gruen A., Van Gool L.: Automatic extraction of man-made objects from aerial and space images (III) (2001)
3. Blaschke T.: Object based image analysis for remote sensing, ISPRS Journal of Photogrammetry and Remote Sensing, Elsevier, vol. 65 (1), pp. 2–16 (2010)
4. Blaschke T., Lang S., Hay G.J.: Object-based image analysis: spatial concepts for knowledge-driven remote sensing applications, Springer Verlag (2008)
5. Chandola V., Banerjee A., Kumar V.: Anomaly detection: A survey, ACM Computing Surveys (CSUR), vol. 41 (3), pp. 1–58 (2009)
6. Glassner A.: Fill'Er Up!, IEEE Computer Graphics and Applications, pp. 78–85 (2001)
7. Gonzalez R. C., Woods R. E.: Digital image processing. London, Pearson Education, 954 pages (2008)
8. Horak Z., Kudelka M., Snasel V.: FCA as a Tool for Inaccuracy Detection in Content-Based Image Analysis, IEEE International Conference on Granular Computing, pp. 223–228 (2010)
9. Lee D., Seung H.: Algorithms for Non-Negative Matrix Factorization, Advances in Neural Information Processing Systems, vol. 13, pp. 556–562 (2001)
10. Li H. T., Gu H. Y., Han Y. S. et al.: Object-oriented classification of high-resolution remote sensing imagery based on an improved colour structure code and a support vector machine, International journal of remote sensing, vol. 31 (6), pp. 1453–1470 (2010)
11. Lillesand, T., Kiefer, R.: Remote Sensing and Image Interpretation, New York, John Wiley and Sons, 724 pages (2000)
12. Mayer H.: Automatic object extraction from aerial imagery–A survey focusing on buildings, Computer vision and image understanding, Elsevier, vol. 74 (2), pp. 138–149 (1999)
13. Mena, JB: State of the art on automatic road extraction for GIS update: a novel classification, Pattern Recognition Letters, vol. 24 (16), pp. 3037–3058 (2003)
14. Pitas I.: Digital image processing algorithms and applications, New York, Wiley, 419 pages (2000)

15. Riedmiller M.: Advanced supervised learning in multi-layer perceptrons–From backpropagation to adaptive learning algorithms, Computer Standards & Interfaces, vol. 16 (3), pp. 265–278 (1994)
16. Umbaugh S. E.: Computer imaging: digital image analysis and processing, London, Taylor & Francis, 659 pages (2005)
17. Yegnanarayana B.: Artificial neural networks, PHI Learning Pvt. Ltd. (2004)

Content-based Retrieval of Compressed Images

Gerald Schaefer

Department of Computer Science
Loughborough University
Loughborough, U.K.
gerald.schaefer@ieee.org

Abstract. Content-based image retrieval allows search for pictures in large image databases without keyword or text annotations. Much progress has been made in deriving useful image features with most of these features being extracted from (uncompressed) pixel data. However, the vast majority of images today are stored in compressed form due to limitations in terms of storage and bandwidth resources. In this paper, we therefore investigate a different approach, namely that of compresseddomain image retrieval, and present some compressed-domain image retrieval techniques that we have developed over the past years. In particular, a method for retrieving images compressed by vector quantisation, that uses codebook information as image features, is presented. Retrieval of losslessly compressed images obtained using lossless JPEG, can be retrieved using information derived from the Huffman coding tables of the compressed files. Finally, CVPIC, a 4-th criterion image compression technique is introduced and it is demonstrated that compressed-domain image retrieval based on CVPIC is not only able to match the performance of common retrieval techniques on uncompressed images, but even clearly outperforms these.

Keywords: content-based image retrieval (CBIR), image compression, compresseddomain image retrieval, vector quantisation, lossless JPEG, CVPIC

Simple Mathematical Models in Biometric Image Analysis

Dr. Khalid Saeed, DSc, PhD, MSc, BSc Engg

Faculty of Physics and Applied Computer Science
AGH University of Science and Technology
Cracow, Poland saeed@agh.edu.pl, khalids@wp.pl
http://home.agh.edu.pl/~saeed/

Abstract. Biometrics is a science that deals with human identification on the basis of our biological features. Therefore, Biometrics belongs to Pattern Recognition and is part of it. Biometric examples are all features we are born with like facial image, finger-prints, iris of the eye, ... or the features we learn in our life like the way we write (signature), the way we walk (gait) or any of the behavioural characteristics. One of the basic steps in the procedure of pattern recognition for the right decision taken with high success rates of identification and verification is the way we furnish the characteristic points of the human biometric images. Once the biometric image is represented by the actual description, easy for implementation in the available popular computing systems, the recognition results will then be more satisfying. The characteristic points should cover all the essential information carried by the selected features that are necessary and in high percentage sufficient for human identification and/or verification.

A general study of all biometric categories will be discussed with examples. The methods of biometric image preprocessing will also be given in order to show the biometric image preparation for classification and recognition. The worked out algorithms with their mathematical approaches and models represented by the feature vectors will be shown.

During the talk, the author will present a method for image description derived from the theory of analytic functions. The original mathematical importance of Toeplitz matrices, which are positive definite, is in the theory of the classical Caratheodory coefficient problem, proved independently by Toeplitz and Caratheodory in 1911. Caratheodory investigated power series which are analytic in the unit circle and have a positive real part in the unit circle. This will take us to Caratheodory and Schur theorems [1], to their modified and developed assertions used in Electric Circuit Theory – Network Synthesis and applied by the author to the Digital Filter Realisation and then to Image processing [2]. The contribution of Toeplitz and his matrices to the subject will also be discussed with examples on the use of the theory in Image description for the sake of their object recognition. The main idea is based on employing the mentioned above classes of analytic functions to build a mathematical model for image description for classification by either the determinants or matrix lowest eigenvalues. Both the matrix lowest eigenvalues and their determinant approaches will be discussed. How to construct the

image feature vector by the aid of the matrix determinants sequence will be shown as a new method of image description. The matrices, in turn, are shown how to be formed from the geometric features of the object image, the object being one of the human biometric feature images or, as a general case in recognition systems, any other object under testing for recognition.

References

1. I. Schur. Methods in Operator Theory and Signal Processing I. Gohberg, ed., Birkhäuser, 1986, 31–89.
2. Saeed K.. A mathematical model on Töeplitz lowest eigenvalues in signal and image processing. Accepted for presentation at IMACS 2011, Morocco 2011.

Revision of Relational Joins for Multi-Core and Many-Core Architectures*

Martin Kruliš and Jakub Yaghob

Department of Software Engineering
Faculty of Mathematics and Physics
Charles University in Prague
{krulis, yaghob}@ksi.mff.cuni.cz
http://www.ksi.mff.cuni.cz/

Abstract. Actual trend set by CPU manufacturers and recent developement in the field of graphical processing units (GPUs) offered us the computational power of multi-core and many-core architectures. Database applications can benefit greatly from parallelism; however, many algorithms need to be redesigned and many technical issues need to be solved. In this paper, we have focused on standard relational join problem from the perspective of current highly parallel architectures. We present comparison of different approaches and propose algorithm adaptations which can better utilize multiple computational cores.

Key words: relational, join, intersection, parallel, multi-core, many-core, GPGPU

1 Introduction

Joins in relational databases have been studied thoroughly since they are one of the most essential operations. Even though the algorithms are not likely to be improved anymore, the implementation details need to be revised every few years as they are sensitive to many aspects of hardware architectures, which are changing constantly.

In the past few years, parallel architectures become available for common users. Central processing units with up to 12 logical cores are occupying mainstream segment of the market and graphical cards containing tens or even hundreds of processing units are present in almost every PC. Furthermore, the field of general purpose computing have encountered several major changes in both software accessibility[1] and hardware design.

Unfortunately, sometimes the parallelism cannot be exploited as easily as hardware manufacturers suggest. There are many concerns that need to be taken into account. In case of multi-core CPUs, there are issues regarding memory

* This paper was supported by the grant SVV-2011-263312 and by the grant agency of Charles University (GAUK), project no. 277911.

[1] A release of OpenCL framework for instance

access. Namely the cache coherency maintenance on multiple cores, bottleneck created by multiple cores accessing memory via single bus, or different memory access latency caused by NUMA [1] factor. In case of many-core GPUs, the memory access problems become even more severe. First problem is that the GPU has its own memory which is connected to the rest of the system via external PCI-Express bus. Second problem is that the caches of a GPU are much smaller than caches of ordinary CPU, therefore access to the memory must be carefully optimized. In addition, the GPU architecture is different to the architecture of CPU as it exploits single instruction multiple data paradigm. In this paper, we will attend described issues and present some solutions.

For the sake of simplicity, we have reduced the problem of join operations, since they have many variations based on its purpose, datatypes, existence of indices, etc. Henceforth, we define join operation as a simple intersection of two key sets. Both sets have unique keys, which are numbers of fixed size (e.g. 32-bit integers). We also assume that the keys are distributed almost uniformly among their domain.

The paper is organized as follows. Section 2 reviews related work. Section 3 summarizes shortly standard serial algorithms for the join problem. Section 4 presents and compares possible parallelization techniques for serial algorithms. Section 5 addresses problems of many-core architectures (such as GPU cards) and revise parallel algorithms from their perspective. Finally, Section 6 presents experimental results and Section 7 concludes.

2 Related Work

Relational joins [2] are one of the most important database operations. There are many papers related to the subject of parallel join processing taking many different views. Liu et al. [3] focused on the pipelined parallelism of multi-join queries. In contrast, we are focusing on accelerating processing of single join query. Lu et al. [4] compared four hash-based join algorithms on a multiprocessor system. Schneider et al. [5] studied join algorithms in share-nothing system. Cieslewicz et al. [6] implemented highly parallel hash join on the Cray MTA-2 architecture. Recently, Changkyu et al. [7] reviewed hash and sort join algorithms from the perspective of modern multi-core CPUs.

New modern many-core architectures, such as GPU [8] become available for programers thanks to GPGPU languages and frameworks such as CUDA [9] and OpenCL [10]. The GPGPU techniques are summarized in survey of Owens et al. [11]. The GPU has already adopted many tasks used in query processing, such as sorting [12][13]. Bakkum et al. [14] implemented SQL query processor accelerated by GPU. Bingsheng et al. [15] studied several types of join algorithms using low level data primitives such as split or gather/scatter implemented on GPUs. We will reflect some of their findings in our work, but we use different approach to the problem that will exploit new hardware features and architectures of today.

3 Serial Joins

3.1 Brief Overview

In this section we will review existing join algorithms shortly, so we can refer to them later. We also pinpoint their strengths and weaknesses in the perspective of current hardware properties.

There are two major approaches to solving join problem. First approach widely used was the merge join. Both joined sets are sorted first and then merged in single pass. Originally, it was designed for systems with small amount of internal memory since the sorting can be achieved using algorithms for external memory and the merge phase requires little additional memory. At present time, the merge join algorithm is used only in special cases. For instance, if the joined sets are (or can be) yielded by previous stages of query execution pipeline already sorted, the sorting phase may be omitted.

As the amount of internal random access memory in computers grew, database systems replaced merge join with hash join algorithm. Hash join stores one of the sets into a hash table and then looks up each item from the second set in the table. If the item is found there, it is included into the result. Hash join principles are used in the state of the art algorithms of today.

Special case of hash join algorithm uses bitmap as a hash table. In case the key domain is sufficiently small, we can create a bit field where each bit corresponds to one item from the domain. In our case, such field requires 2^{32} bits (512 MB). When a bit is set, the corresponding key is present in the hashed set and vice versa.

3.2 Considering Hardware Aspects

There are two most important aspects of current CPU architectures that affects join performance [7]:

- CPU caches,
- and virtual memory translation.

CPU uses two or three level caches to reduce memory access latency. The size of the cache is much smaller than size of main memory, thus accessing large memory areas randomly is not very efficient. Furthermore, the processor employs prefetch mechanisms which tries to identify memory that is likely to be needed soon and load it into the cache in advance. These mechanisms work on their best if the memory is accessed sequentially.

Second important issue is virtual memory translation. On IA32 architecture, the translation is performed through page tables [16]. Depending on virtual address space and some other details, the address is translated by looking up records in 3 or 4 tables. Therefore, each virtual memory access leads into 4 or 5 physical memory accesses. In order to reduce this overhead, modern CPUs implement TLB cache for translation. However, TLB cache is limited in its size, thus if we access larger amount of virtual pages randomly, the TLB misses are more frequent and memory latency rises.

In the light of current CPU architectures, we can redesign both merge and hash join algorithms. The sorting algorithm, which takes more than 90% of merge join time, can be optimized [17] and the hash join can compact its memory access pattern by introducing partitioning techniques [7] that we call bucketing. These optimization are well beyond the scope of this paper, but we will exploit bucketing for parallel reasons later.

4 Multi-Core Implementation

In this section, we will consider multi-core CPU architectures to increase performance of join algorithms. We will examine strengths and weaknesses of merge and join algorithms from the parallel point of view and then improve them with the bucket partitioning.

4.1 Direct Approach

Merge Join can be parallelized quite easily as the sorting algorithms are known to scale well [17]. The final merge can be parallelized as well by separating one of the sorted sets into fragments and processing each fragment concurrently. The only complication is that we need to find corresponding fragments in the second set so both fragments can be merged; however, these fragments are easily identified by simple application of binary search algorithm.

Hash join does not scale as well as merge join. We need to synchronize access to hash map (or the bitmap) by some kind of locking mechanism or atomic operations. The synchronization creates bottleneck of the algorithm, therefore the speedup is rather small as we can see in Section 6.

4.2 Bucketing Exploitation

The best way to solve any problem in parallel is to create independent tasks so that they can be processed by multiple threads without explicit synchronization. In case of join algorithms, we can employ the bucketing principle. We use hashing function to separate data into buckets and each bucket can be processed by a different thread. For the sake of simplicity, we always divide the set among 2^k buckets using upper k bits from the key. We can use more sophisticated hashing functions in case the key distribution is distorted in any way.

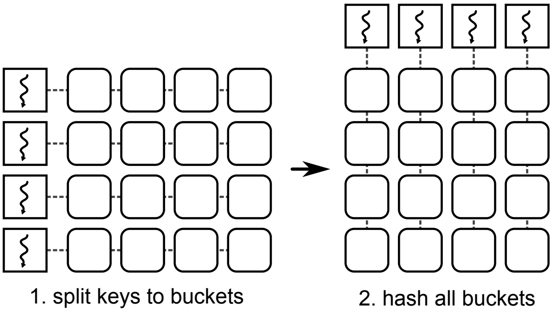
Divide And Conquer. One of possible solutions is based on divide and conquer programing paradigm. It requires that the hashing function can be incrementally refined. In our case, we can simply use more bits from the key for more fine grained bucketing. The algorithm is similar to QuickSort [18]. In the first pass it takes the array representing set of keys and reorder them so that keys with uppermost bit equal to 0 are on the left and keys starting with 1 are on the right. Then both parts of the array are processed recursively using two uppermost bits, etc. The recursion stops in predefined depth or when a bucket size decrease below predefined threshold. This approach has several advantages.

- The algorithm does not require any additional memory. We can perform splitting phases in place as the QuickSort does.
- Recursive calls can be processed by different threads since they work on disjoint ranges of the array.
- We can use different grain for different buckets, thus achieving more balanced workload even if the keys are not distributed homogeneously.
- Each splitting phase works sequentially so it benefits from caches.

After partitioning both sets into buckets, these buckets are processed concurrently. A simple hash join algorithm (using bitmap as a hash table) is used locally. Thanks to the partitioning, we require only 2^{32-k} bits for the bitmap.

Two-Pass Bucketing. Previous approach is quite effective, but it has two flaws. It takes a while before recursion reaches sufficient depth to fully exploit parallelism and we need to process each item $\mathcal{O}(k)$ times. We can achieve bucketing by one-pass algorithm if we put items directly into their buckets. The problem is that if we want to run such algorithm concurrently, access to these buckets needs to be synchronized, thus creating additional overhead. We propose lock-free adaptation, which requires two passes, but it can run concurrently on all available cores from the beginning.

We create a matrix of buckets $T \times b$, where T is number of threads available and b is number of buckets, so that each thread has its own set of buckets. We also expect that $T \leq b$, otherwise the following stages of the algorithm cannot fully utilize all the threads. For the sake of simplicity, we describe the algorithm for $T = b$.



In the first phase, each thread takes its equal share of input set and divide it into its local buckets. When the keys are scattered to the buckets, we prepare vector of b hash tables (in our case bitmaps). Each thread takes one column of the matrix (i.e. group of buckets containing keys with the same prefix of length k) and save them in the corresponding hash table. Threads can access hash tables without locking since each thread works on a separate bucket group and each group has its own table. Finally, the second key set is processed concurrently. Proper bucket is determined for each key in the set and the key is looked up in its hash table. No synchronization is required as the hash tables are used only for reading.

5 Adaptation for Many-Core Architectures

Before we start designing join algorithms for many-core GPUs, we recall the most important facts about the architecture first. These facts need to be considered carefully since they affect performance significantly.

5.1 GPU Architecture

Kernel Execution. There are two main concerns about GPU architecture. The first is rather specific program execution. Portions of code, which are to be executed on GPU, are called kernels. Kernel is a function that is invoked multiple times – once for each working thread. Each thread gets the same set of calling arguments and a unique identifier from 0 to $N - 1$ range where N is the number of invoked threads. The kernel uses thread identifier to select proper part of the parallel work. The thread identifiers may be also organized in two or three dimensional space for programmers convenience.

The thread managing and context switching capabilities of a GPU are very advanced. Therefore, it is usually better to create huge amount of threads even if they execute only a few instructions each. More threads are better for load balancing and fast context switching can be used to hide latency of accessing global memory.

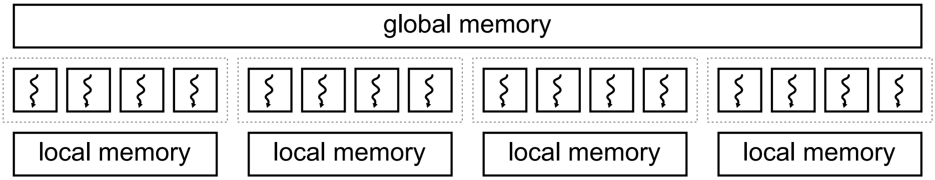


Fig. 1. Thread organization

Threads are aggregated into small bundles called groups. A group usually contains tens to hundreds threads which execute the kernel in SIMD[2] or virtual

[2] Single Instruction Multiple Data

SIMD fashion. Every thread executes the same instruction, but it has its own set of registers, thus working on different portion of data. SIMD suffers from branching problem – different threads in the group chose different branches of 'if' statements. To execute conditional code properly, all branches must be executed by all threads and each thread masks instruction execution according to local result the condition. On the other hand, SIMD approach eases synchronization within the group and threads can collaborate through shared local memory.

Finally, we have to point out that most of the graphic hardware is not capable of executing different kernels simultaneously, even if they do not occupy all the processing units. The only architecture currently capable of simultaneous kernel execution is the newest NVIDIA Fermi [8].

Memory Organization. We have four different memory address spaces to work with when programming GPUs:

- host memory,
- global memory,
- local memory,
- and private memory.

The *host memory* deserves extra attention, since it is not directly accessible from processing units. Input data needs to be transferred from host memory to graphic device memory and the results needs to be transferred back to host memory when the computation terminates. Furthermore, these transfer use PCI-Express bus, which is rather slow (in comparison with internal memory buses).

The *global memory* is directly accessible from GPU cores. Input data and computed results of a kernel are stored here. The global memory has high both latency and bandwidth. In order to access the global memory optimally, threads in one group are encouraged to use *coalesced loads.* Coalesced load is performed when all the threads in the group loads or stores continuous block of aligned memory, so that each thread processes one 4-byte word (see Figure 2).

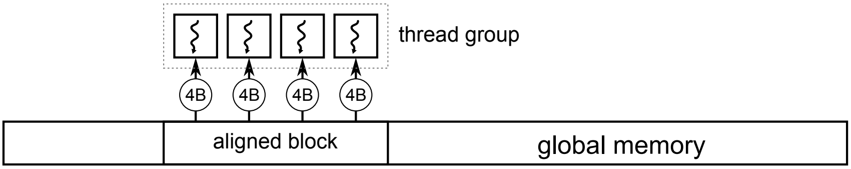


Fig. 2. Coalesced load

The *local memory* is shared among threads in one group. It is very small (in order of tens of kB) but almost as fast as GPU registers. Local memory can play role of program-managed cache for global memory, or the threads may store partial results in here while they cooperate on a task. The memory is separated into several (usually 16) banks. Following two 4-byte words are stored in following two banks (modulo number of banks). When two threads access the same bank (except if they read the same address), the memory operations are serialized.

Finally, the *private memory* is private to each thread and corresponds to processor registers. Private memory size is very limited (tens to hundreds of words), therefore it is suitable just for few local variables.

5.2 Algorithm Design

There are three different approaches to implementing join algorithm on GPU. First exploits the GPU power for sorting. Sorting algorithms for GPUs have been already thoroughly studied [12][13], but our preliminary results indicated that the sorting operation takes too much of precious time and the join algorithm can be implemented better. Second is to implement bucketing on GPU and create buckets small enough so they fit to local memory. This is also promising way; however, fine grained bucketing takes also quite large amount of time.

In our work, we examined the third possibility – applying hash join using bitmap as hash table on GPU. Unfortunately, the bitmap would require continuous memory block of 512 MB, which cannot be allocated on present GPU hardware due to some technical limitations. We have applied lightweight bucketing in order to reduce bitmap size so it would fit GPU memory. The algorithm is designed as follows:

```
for both input sets do
  calculate histogram (number of items in each bucket)
  perform prefix sum on the histogram (compute starting offsets)
  split keys into the buckets
end for
for each bucket do
  prepare empty bitmap
  for $\forall x$ of the first set in the bucket do
    set bit corresponding to $x$ to 1
  end for
  for $\forall y$ of the second set in the bucket do
    if bit corresponding to $y$ is 1 then
      include $y$ to the result
    end if
  end for
end for
```

The histogram is computed in highly parallel fashion using atomic increment operations. The splitting algorithm works similarly like the histogram computation, but it uses local memory as output cache, so that the data are written to global memory in larger blocks.

We have chosen total size of 16 buckets for the current hardware. This way the size of bitmap required for hashing is reduced to 32 MB which is feasible.

When the bitmap is being filled, a thread is created for each item x in the bucket. Data are written using atomic OR operation, thus all conflicts are avoided. After they complete, a thread is created for each key y. Threads in one group uses local memory as a cache for keys that has been included into the result. When the local cache is full, it is spiled into the global memory using atomic add operation to allocate position in output buffer.

6 Experimental Results

6.1 Methodology, Hardware, And Testing Data

Each test was performed 10 times and presented values are the average of measured times. All times were well within ±10% limit from the presented value.

Scalability tests were performed on Dell M910 server with four six-core Intel Xenon E7540 processors with hyper-threading (i.e. 48 logical cores) clocked at 2.0 GHz. Server was equipped with 128 GB of RAM organized as 4-node NUMA. GPU tests were conducted on a common PC with six-core Intel Core i7 870 CPU with hyper-threading clocked at 2.93 GHz. The machine was equipped with 16 GB of RAM and one NVIDIA GTX 580 GPU card with 512 CUDA cores and 1.5 GB of RAM. A RedHat Enterprise Linux (version 6) was used as operating system on both machines.

We use three data set for testing – pairs of $4M$, $8M$, $16M$ key sets. Each set was generated randomly with uniform distribution over the 32-bit key domain. We did not use larger data as they would not fit to the GPU memory.

6.2 Scalability on Multi-Core CPUs

First, we test scalability of CPU multi-core algorithms. Horizontal axis shows number of active threads and vertical axis represents time in ms. The *merge* denotes merge join algorithm, *hash* is hash join with bitmap hash table employing atomic operations for synchronization, *split* stands for divide and conquer recursive bucketing algorithm, and finally *bucket* represents two-pass bucketing algorithm.

Figure 3 shows several things. First, that the algorithms does not scale very well beyond 8 cores. This is most likely caused by relatively low memory bandwidth. As multiple cores share the same memory controllers and caches, they slow down each other. Second, the best algorithm is obviously bucketing algorithm as we expected [7]. Finally, even though the merge join algorithm does not perform very well on single core, its scalability makes him worthy adversary on eight and more cores.

Figures 4 and 5 shows how the algorithms perform on smaller datasets. The merge join is more suited for smaller sets as it can better utilize CPU caches and TLB. On the other hand, hash join performs worse on smaller data sets as it requires initialization of large bitmap which size is relative to the size of key domain, not the size of joined sets.

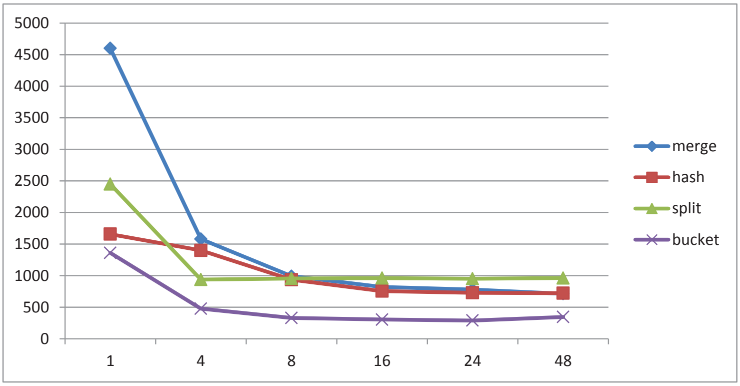


Fig. 3. Join of two sets, 16M keys each

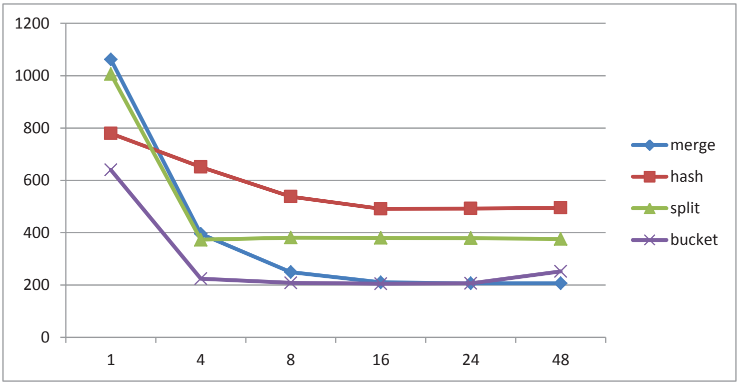


Fig. 4. Join of two sets, 4M keys each

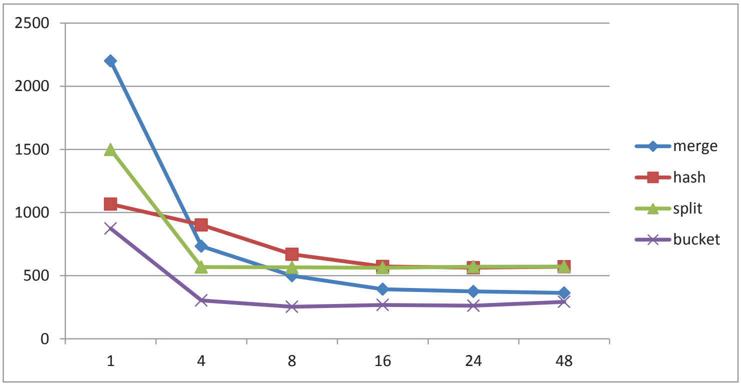


Fig. 5. Join of two sets, 8M keys each

6.3 Performance on GPU

We have compared our GPU algorithm to both serial and parallel algorithms on CPU. The vertical axis represent times in ms. The *hash(1T)* denotes hash join algorithm executed on single thread, *bucket(12T)* is two-pass bucketing algorithm executed on 12 cores, *GPU comp.* stands for GPU algorithm (only computation time), and finally *GPU total* represents times of GPU computation including data transfers between host and device memory. The *hash(1T)* and *bucket(12T)* were selected as best serial and parallel representatives on Core i7 CPU.

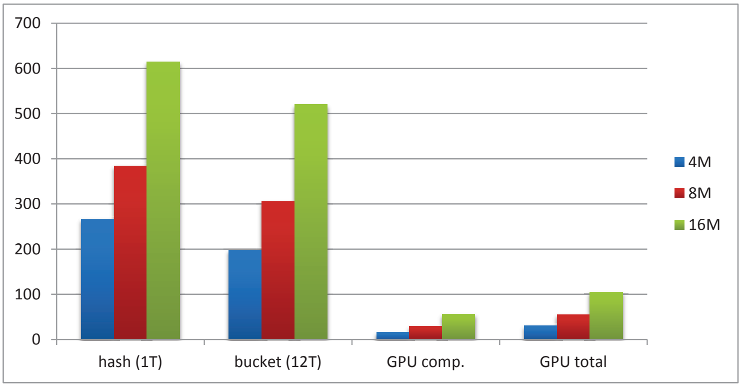


Fig. 6. Comparison of GPU and CPU algorithms

The GPU algorithm runs more than 10× faster than fully parallelized bucketing algorithm running on 12 cores, and still more than 5× faster if we take the time for data transfers into account. We can also observe that the transfer of data to the graphic card and back takes almost the same amount of time as the computation itself.

7 Conclusions

This paper presents comparison of join algorithms and their modifications in the perspective of multi-core CPUs. Partitioning of joined sets into buckets improves performance significantly and it can be easily adopted for lock-free concurrent processing. We have also presented an adaptation of join algorithm that works well on GPU many-core hardware and achieve more than 10× speedup to the CPU parallel algorithms.

There is much work to be done still. We are going to compare our GPU algorithm with other possible implementations and make more thorough analysis of them. We are also planing to compare our join algorithm with standard in-memory database systems. In our future work, we will focus on exploiting many-core GPU hardware to accelerate common database operations and index searching.

References

1. NUMA: Non-Uniform Memory Architecture. http://en.wikipedia.org/wiki/Non-Uniform_Memory_Access
2. Mishra, P., Eich, M.: Join processing in relational databases. ACM Computing Surveys (CSUR) **24**(1) (1992) 63–113
3. Liu, B., Rundensteiner, E.: Revisiting pipelined parallelism in multi-join query processing. In: Proceedings of the 31st international conference on Very large data bases, VLDB Endowment (2005) 829–840
4. Lu, H., Tan, K., Shan, M.: Hash-based join algorithms for multiprocessor computers. In: Proceedings of the 16th International Conference on Very Large Data Bases, Morgan Kaufmann Publishers Inc. (1990) 198–209
5. Schneider, D., DeWitt, D.: A performance evaluation of four parallel join algorithms in a shared-nothing multiprocessor environment. ACM SIGMOD Record **18**(2) (1989) 110–121
6. Cieslewicz, J., Berry, J., Hendrickson, B., Ross, K.: Realizing parallelism in database operations: insights from a massively multithreaded architecture. In: Proceedings of the 2nd international workshop on Data management on new hardware, ACM (2006) 4
7. Kim, C., Kaldewey, T., Lee, V., Sedlar, E., Nguyen, A., Satish, N., Chhugani, J., Di Blas, A., Dubey, P.: Sort vs. Hash revisited: fast join implementation on modern multi-core CPUs. Proceedings of the VLDB Endowment **2**(2) (2009) 1378–1389
8. NVIDIA: Fermi Architecture. http://www.nvidia.com/object/fermi_architecture.html
9. NVIDIA: CUDA. http://www.nvidia.com/object/cuda_home_new.html
10. Khronos: OpenCL – The open standard for parallel programming of heterogeneous systems. http://www.khronos.org/opencl/
11. Owens, J., Luebke, D., Govindaraju, N., Harris, M., Kruger, J., Lefohn, A., Purcell, T.: A Survey of General-Purpose Computation on Graphics Hardware. In: Computer graphics forum. Volume 26., Wiley Online Library (2007) 80–113
12. Sintorn, E., Assarsson, U.: Fast parallel GPU-sorting using a hybrid algorithm. Journal of Parallel and Distributed Computing **68**(10) (2008) 1381–1388
13. Greß, A., Zachmann, G.: GPU-ABiSort: Optimal parallel sorting on stream architectures. In: Proceedings of the 20th IEEE International Parallel and Distributed Processing Symposium, Citeseer (2006) 45
14. Bakkum, P., Skadron, K.: Accelerating SQL database operations on a GPU with CUDA. In: Proceedings of the 3rd Workshop on General-Purpose Computation on Graphics Processing Units, ACM (2010) 94–103
15. He, B., Yang, K., Fang, R., Lu, M., Govindaraju, N., Luo, Q., Sander, P.: Relational joins on graphics processors. In: Proceedings of the 2008 ACM SIGMOD international conference on Management of data, ACM (2008) 511–524
16. Page Tables. http://en.wikipedia.org/wiki/Page_table
17. Chhugani, J., Nguyen, A., Lee, V., Macy, W., Hagog, M., Chen, Y., Baransi, A., Kumar, S., Dubey, P.: Efficient implementation of sorting on multi-core SIMD CPU architecture. Proceedings of the VLDB Endowment **1**(2) (2008) 1313–1324
18. Hoare, C.: Quicksort. The Computer Journal **5**(1) (1962) 10

Author Index