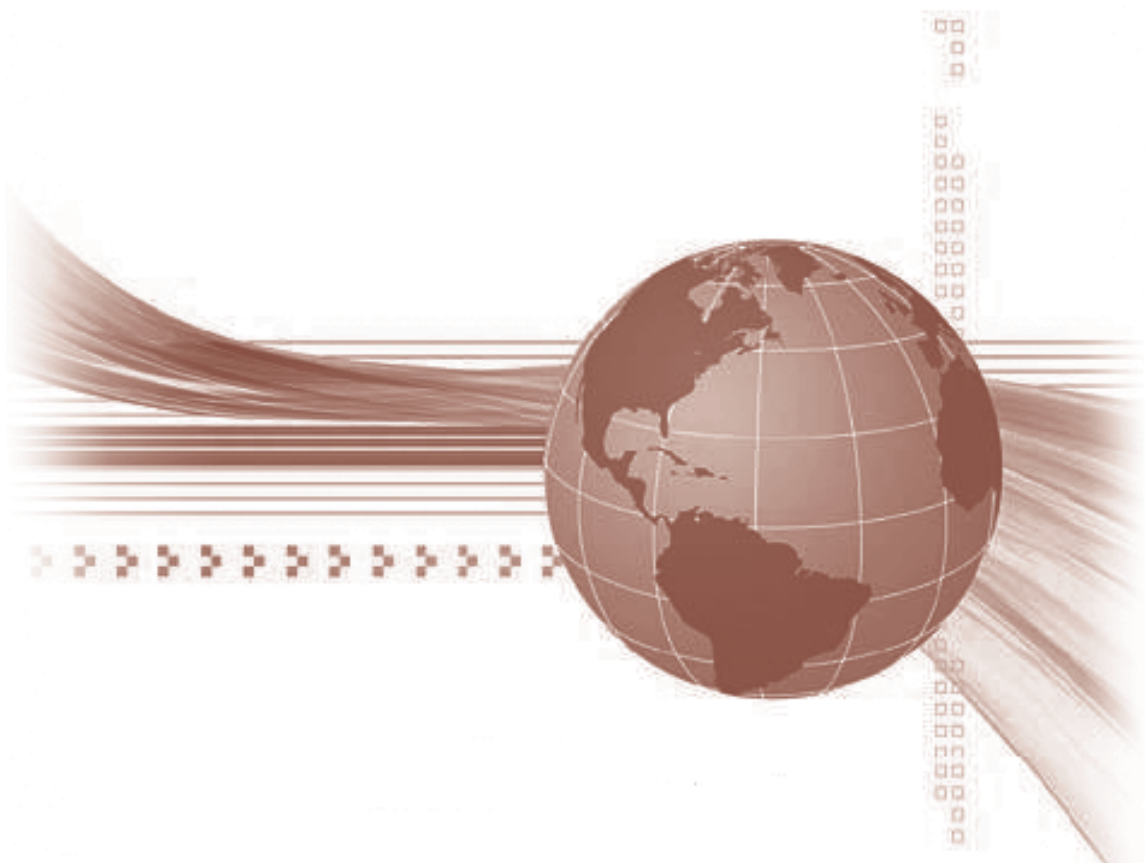
STUDIA UNIVERSITATIS
BABEŞ-BOLYAI



INFORMATICA

1/2014

STUDIA

UNIVERSITATIS BABEŞ-BOLYAI
INFORMATICA

No. 1/2014

January - June

EDITORIAL BOARD

YEAR **Volume 59 (LIX) 2014**
MONTH **JUNE**
ISSUE **1**

S T U D I A
UNIVERSITATIS BABEŞ-BOLYAI
INFORMATICA

1

EDITORIAL OFFICE: M. Kogălniceanu 1 • 400084 Cluj-Napoca • Tel: 0264.405300

SUMAR – CONTENTS – SOMMAIRE

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LIX**, Number 1, 2014

SYN!BAD: A SYNONYM-BASED REGULAR EXPRESSION EXTENSION FOR KNOWLEDGE EXTRACTION TASKS

OVIDIU ŞERBAN

Abstract. This paper focuses on presenting Syn!bad, a synonym-based regular expression language which can be used for basic Knowledge Extraction tasks, such as the Natural Language Understanding components for Dialogue Management. The language offers a simple syntax for complex matching processes, which brings a new solution for the input variability problem, often encountered when dealing with natural language. The language provides various special tokens to match synonym-based expressions free context variables and generic Part of Speech tokens.

1. Introduction

In the field of Natural Language Processing (NLP), manipulating data patterns has been a very successful approach. These patterns are the key to "understanding" the natural language, by grouping similar elements by their functionality. The evolution of automata theory leads to a new language to represent these patterns: the Regular Expressions [16]. There are many standards for Regular Expressions, leading to various implementations [7]. We will focus our discussion on the basic (BRE) and extended (ERE) features of Regular Expressions, because of their frequent usage.

From the task oriented algorithms perspective, regular expressions are very popular in NLP systems, either used alone or in preprocessing tasks for mixed approaches. Most of the sentence tokenization algorithms are based on regular expressions [8, 2]. Moreover, more complex tasks, such as Part-of-Speech Tagging (POS) [13] or Named Entity Recognition (NER) [10] use them as well. Recently, regular expressions are employed to process emoticons [11] for Sentiment Analysis Applications. In general, regular expressions are used in applications where robust but simple algorithms need to be employed

Received by the editors: December 1, 2013.

2010 *Mathematics Subject Classification.* 68T35, 68T50.

1998 *CR Categories and Descriptors.* H.5.2 [**Information Interfaces and Presentation**]: User Interfaces – *Natural Language*; F.1.1 [**Computation by Abstract Devices**]: Models of Computation – *Automata*.

Key words and phrases. Knowledge Extraction, Dialogue Systems, Regular Expressions.

as part of a preprocessing strategy [5]. From the application perspective, the regular expressions have been used successfully in spam detection and filtering [3] or more generic data mining techniques [17].

Syn!bad is an extended regular expression language, for usage mainly in Natural Language Processing (NLP) applications. It uses the extended POSIX Regular Expression [1] structures among others, more specific to NLP domain. Sometimes, the learning phase for the detection algorithms requires a feature extraction method. The knowledge extraction methods, involved in the Natural Language Understanding Process, use similar techniques to detect key concepts to be used in the Dialogue Management process. We propose this language to simplify the construction of these patterns.

The name, Syn!bad (also written: Synnbad, with double nn, instead of n!) is an acronym of *Synonyms [are] not bad*. This suggests that the main concepts of Syn!bad are centred among synonyms processing, using different dictionaries.

Synonyms are independent structures, grouped in different sets, by their meaning. The most common grouping currently known is the WordNet[1] synsets [9], which consists in grouping different words according to their semantics and part of speech. The basic unit, a synset (a set of synonymous words which refer to a common semantic concept), has a unique id, which permits an easy retrieval.

Syn!bad is available both as an independent library and as a component of the AgentSlang platform [14]. Nevertheless, Syn!bad is intended to be a platform and language independent library that can be implemented and distributed on its own. We present the language in the scope of basic knowledge extraction for Interactive Systems (IS), but this library can be extended to document classification, summarisation, topic extraction, etc.

In dialogue management, knowledge extraction or affect detection, building a set of patterns to extract the information simplifies the complexity of any system. Moreover, it gives a tool set flexible enough to process any data. Appendix A provides a formal view over the language, by presenting the BNF Grammar definition of Syn!bad .

This article is organized as following: we start with a brief presentation of our context, related to the Interactive System domain. We continue with the description of the system, which includes several implementation details. At last, we conclude with a discussion about the current paper.

[1]WordNet is a commonly known lexical database for English language.

2. Context of the problem

In the field of Interactive Systems (IS), the knowledge extraction process is usually slowed down by the complexity of the rules describing a certain concept. Using regular expressions is an alternative, but in certain situations, composing rules for all the cases is impossible. Another approach is to group certain structures while making them more generic. For instance, instead of using a regular expression for matching the following sentence: `Bob can I have your phone`, one could use `<name> can I <verb> your <object>`. By using regular expressions, the variable structures are already supported by certain implementations.

When adding restrictions to the matched variables the problem becomes more difficult, especially in the case of `<verb>` and `<object>`. To our knowledge, the syntax of matching only variable structures while having a certain part of speech is not supported by any regular expression implementation.

A more complex situation is given by placing a synonymic relation restriction on the matched item. In our previous example, we would like to extract only the objects being synonyms of the word *phone*. The synonyms usually introduce a certain fuzziness into a decision, since not all the meanings of a polysemantic word match the context of a given pattern. In this scenario, a certain restriction can be modelled, by adding a part of speech restriction on the word. For example, the word *phone* has multiple meanings, such as *the action of calling someone*, when employed as a verb or *telephone (object)*, when used as a noun. When matching a synonym of *telephone* with our rule, we can restrict this to only *nouns*, in which case the words *call* or *ring*, as verbs, are not matched.

Another situation is dealing with variable concepts, restricted or not by a certain format. Given the following phrase: `1000$ for a phone ???`, we observe that the amount and the target object may be *variable* but also a very important parameter involved in the decision process. Nevertheless, several restrictions can be added for the amount and the object. For example, the amount clearly needs to be a number, which can be an annotation done previous to the decision and the object can be either a generic object or a synonym of the word "phone". In the end, for all our examples, the style and format of the pattern are decided by a human coder.

3. Preliminaries

The ERE, as an extension of BRE, introduces several basic operators:

- The simple terminal matching operator, which can be a character or a sequence of characters: `aabbac`, `[a-z]bce[0-9]`. In these examples,

the matching process is successful if the sample matches either the exact sequence `aabbac` or it starts with any low-case character, continues with the `bce` sequence and ends with a digit, for the case of the next sequence.

- The alternative operator (`|` which is synonym with the **or** keyword): `seq1|seq2`. The matching process is successful if any of the sequences 1 or 2 is matched.
- The zero or one time operator (`?`), which describes an optional token.
- The zero or many times operator (`*`), which allows the token multiplication during the matching process.

Several other operators are either specific to character sequence matching (`^` - for the negation of character interval) or are equivalent to a combination of multiple operators: `seq+` $\Longleftrightarrow$ `seqseq*`.

In our previous examples, we used the synonyms of the word *phone*. In fact, a very popular structure for describing the synonym grouping is given by *synsets* (Synonym Sets), as the core part of WordNet [9]. These synsets consist in a group of synonym words, which have been annotated according to their Part of Speech (POS) and have a gloss attached to describe their meaning. For an ease of use in Computer Science applications, these synsets have a unique alphanumeric index attached. The following example is the synset of the word *phone*:

04401088: (n) telephone, ***phone****, telephone set (electronic equipment that converts sound into electrical signals that can be transmitted over distances and then converts received signals back into sounds)*

where *04401088* is the synset identification number, *(n)* states that this synset contains nouns, followed by the synonym list and the last part is the gloss. Usually, the synset id is represented as a combination between the part of speech and the identification number, becoming in this case `n#04401088` or simply `04401088n`.

Another important concept used in our work is the Part of Speech (POS) tag, which corresponds to the lexical class of each word. The process of annotating a word with its specific tag is usually called Part-of-Speech Tagging and it cannot be done on an independent word, but on a whole phrase or context. Various annotation models exist, each depending on one or multiple POS Tag sets, which are usually language dependent, since not all the languages have the same lexical classes. For English, one of the most popular Tag Set available is the Penn Part-Of-Speech Tag System [12]. For French one of the most common used tag set is the TreeTagger Part-of-Speech Tags

[15], while for Central and East European languages (including Romanian) the MULTEXT-East [6] is frequently used. For the purpose of this article, we will focus our examples on the Penn POS Tag Set, with the observation that our proposition is intended to be language independent.

We continue this paper with a simple practical example of Syn!bad syntax, which will allow us to introduce all the concepts of the proposed language.

4. The Syn!bad Extension

4.1. **A practical example.** Given the previous context, we propose a first example of a Syn!bad pattern.

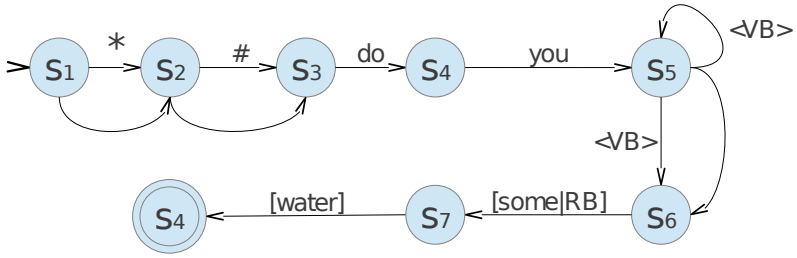


Figure 1. A Syn!bad example, presented as an automaton

Based on the rules described above, we compile the following Syn!bad pattern, which also contains most of the features of the language:

```
$name <#*>? do you <VB*>*  [some|RB*] [water#object]
```

- `$name` item represents a context free variable, which matches any single word and retrieve it as the *name* variable.
- `<#*>?` is an optional token that can match any punctuation mark. Moreover, the *#** represents a generic part of speech group matching punctuation marks.
- *do* and *you* are precise words matched by this expression.
- `<VB*>*` is a none-or-many token matcher, which restricts the element to match only a selected part of speech, in this case a verb.
- `[some|RB*]` represents a matcher for a synonym of the word *some*. Moreover, a restriction over the part of speech is added, which matches only adverbs.
- `[water#object]` this token is similar to the previous one, but a synonym of the word *water* is matched and the word is stored into the *object* variable.

In order to describe the whole matching process, the following sentence is given: `Ovidiu , do you want any aqua`

The result of the matching is: $\$name \leftarrow Ovidiu$ and $\#object \leftarrow aqua$, while `<#*>?` matches the comma mark (,), `<VB*>*` matches the single verb *want* and any is matched by the token `[any|RB*]`.

4.2. **Implementation Details.** The Syn!bad language has two levels, one is related to the grammar model, presented in the Appendix A. The second level concerns the implementation of this language, as an extension to the current capabilities of our knowledge extraction platform.

The patterns are compiled into a Deterministic Finite Automaton (DFA), completely written from scratch in Java language. We choose this representation since the DFA offers superior matching speed for a linear decision. Cox [4] presents a series of experimental arguments to sustain our choice.

The Deterministic Finite Automaton (DFA) is a type of automaton, where each state, for a given input, has at most one new state leading from the previous one. This makes the navigation through the states easier, since the possibility of exploration is always reduced to only one state or none. The finite status is given when our machine reaches one of the terminal states and the finish condition is fulfilled.

The simple token matchers use a simple word equality operator, whereas the others require more complex operators, such as part of speech matchers and synonym intersection.

Concerning the part of speech restrictions, we propose two different types of labels. One is more strict, as recommended by the Penn Part-Of-Speech Tag System [12], which contains 45 different labels. The second is a functional grouping of the first system, called Generic POS, and contains only 5 labels:

(1) `#*` groups all the punctuation marks into one single category: `$ # . , : ( ) " '`
(2) `VB*` groups all the verb tags: `VB, VBD, VBG, VBN, VBP, VBZ`
(3) `RB*` groups all the adverb tags: `RB, RBR, RBS`
(4) `NN*` groups all the noun tags: `NN, NNS, NNP, NNPS`
(5) `JJ*` groups all the adjective tags: `JJ, JJR, JJS`

The synonyms are currently extracted from the WordNet dictionary [9]. We use the synset identifiers, provided by WordNet, restricted by Part of Speech, when necessary. WordNet provides an index already split by part-of-speech, which makes the restriction conditions much easier to fulfil.

All the part-of-speech restrictions, synonyms and variable names are stored as a matching token, making possible to model our automaton as a DFA. All the tokens of a pattern are stored as a linked multi-list.

For each state, we assign a priority to each token, which makes the matcher decision even more simple. The top priority is assigned to the *optional token*,

just before the *mandatory element*. This is done because it is more important to match an optional item, when possible, rather than a mandatory one. The process cannot continue without matching all the mandatory elements, therefore since the optional item can be skipped easily, it is important to match them before the mandatory items. The last priority is assigned to a consumer item, which is either a *skip* item or a *global variable* (a structure labelled `$name`). A *skip* is an element with the lowest priority assigned, which matches everything and it is used to define matching spaces. The current implementation uses a *skip* of 2 items defined by default.

Once the patterns are compiled, each one of them has an identifier assigned. These are not mandatory to be unique, and in certain situations it can be useful to have duplicate identifiers, such as in the case of having polysemic expressions. For instance, the patterns: (`hello`) and (`hi there`), can have the same id (`id=greeting`), since both represent different forms of greetings.

When a pattern is matched, its identifier is returned, along with all the variables matched. The variables could be global: defined as `$name`, or local: `#name` which are defined by the part-of-speech or synonym matching tokens. For instance, the pattern (`hello $name`) matches the first word that comes after `hello` and stores it in the `$name` variable, whereas the pattern `hello <NN*#name>*` matches the first *noun* that follows the word `hello` and stores it in the variable `#name`. In fact, the `$name` variable is matching any word or punctuation mark, whereas `#name` variable stores the content matched by a specific token: part-of-speech or synonym.

4.3. **Syn!bad Pattern Styles.** Patterns, among the variable retrieval feature, have another level of static labels, named styles. A style is represented by a collection of pairs (label, value) assigned to each pattern. The functional value of this feature is represented by the possibility to manually assign a second level of annotation to a certain matcher. The label space is defined on the whole matcher container (all the pattern matchers added on the same list), and the label space is sparse, as well. When a matcher does not have a label defined, an '*' is automatically assigned to any undefined value.

To introduce the *styles*, we present a short example of this functionality. Table 1 defines three different patterns, each one having different styles assigned. Styles are comma separated, defined as a label=value pair.

The pattern *p1* has two values assigned to the styles relation=familiar and rudeness=high, *p2* defines a value just for rudeness=low therefore the relation becomes `*`, *p3* has the *polite* value assigned to the relation. Table 2 summarises these results.

Pattern	**ID**	**Style**
what do you want ?	p1	relation=familiar, rudeness=high
what can i do to help you ?	p2	rudeness=low
if i may ask , how could i help you ?	p3	relation=polite

TABLE 1. Syn!bad pattern examples, using the style definition features

Style	**ID**		
	p1	p2	p3
relation	familiar	*	polite
rudeness	high	low	*

TABLE 2. The values assigned to each style according to the pattern definitions from Table 1

In another context, the styles could be used for other applications, such as sentiment detection. Given the following patters, presented in Table 3, we could observe that the pattern **p1** and **p2** are annotated with negative or positive labels, while **p3** is neutral, therefore no sentiment will be associated. During the matching process, the style for **p3** will be equal to *.

Pattern	**ID**	**Style**
This `$object` is broken !	p1	sentiment=negative
This `$object` is very good !	p2	sentiment=positive
This is a `$object` !	p3	

TABLE 3. Sentiment annotation used as Syn!bad styles

The usage of styles is not mandatory, but offers another level of granularity for the knowledge extraction model. The styles offer a complementary function for variable extraction and in case of large pattern databases, it also provides more information for the dialogue selection models and dialogue generation components.

5. Conclusion

Syn!bad is an extension to the POSIX regular expression language that employs special elements useful for NLP applications. These elements are synonymic or part-of-speech expressions that can be combined with regular word items. The patterns can be grouped into semantic clusters and have various styles assigned, which makes the matching process useful for knowledge extraction and dialogue management. In fact, this language is a critical part of the AgentSlang Platform [14], ensuring the Natural Language Understanding function of the system.

Appendix A. Syn!bad: BNF Language Specification

The BNF Grammar of the Syn!bad syntax is defined as following:

⟨*expression*⟩ ::= ⟨*token*⟩ '␣' ⟨*expression*⟩ | ⟨*token*⟩

⟨*token*⟩ ::= ⟨*pattern*⟩
| ⟨*pattern*⟩ '`*`'
| ⟨*pattern*⟩ '`?`'
| ⟨*pattern*⟩ '{' ⟨*number*⟩ '`,`' ⟨*number*⟩ '}'

⟨*pattern*⟩ ::= ⟨*word*⟩
| '`<`' ⟨*POS_Structure*⟩ '`>`'
| '`[`' ⟨*synonym*⟩ '`]`'
| '`$`' ⟨*variable*⟩

⟨*POS_Structure*⟩ ::= ⟨*POS*⟩ ('`#`' ⟨*variable*⟩)?

⟨*POS*⟩ ::= ⟨*PennPOS*⟩
| ⟨*GenericPOS*⟩

⟨*synonym*⟩ ::= ⟨*word*⟩ ('|' ⟨*POS*⟩)? ('`#`' ⟨*variable*⟩)?

⟨*number*⟩ ::= [1-9] [0-9]*

⟨*word*⟩ ::= [a-z]+

⟨*variable*⟩ ::= [a-z0-9]+

⟨*PennPOS*⟩ ::= 'JJ' | 'RB' | 'DT' | 'TO' | 'RP' | 'RBR' | 'RBS' | 'LS'
| 'JJS' | 'JJR' | 'FW' | 'NN' | 'NNPS' | 'VBN' | 'VB' | 'VBP'
| 'PDT' | 'WP$' | 'PRP' | 'MD' | 'SYM' | 'WDT' | 'VBZ' | '¨'
| '#' | 'WP' | '’' | 'IN' | '$' | 'VBG' | 'EX' | 'POS' | '('
| 'VBD' | ')' | '.' | ',' | 'UH' | 'NNS' | 'CC' | 'CD' | 'NNP'
| 'PP$' | ':' | 'WRB'

⟨*GenericPOS*⟩ ::= '#*' | 'VB*' | 'RB*' | 'NN*' | 'JJ*'

References

[1] V. Alfred. Algorithms for finding patterns in strings. *Handbook of Theoretical Computer Science: Algorithms and complexity*, pages 255 – 300, 1990.

[2] Steven Bird, Ewan Klein, and Edward Loper. *Natural language processing with Python*. O'Reilly Media, 2009.

[3] Eric Conrad. Detecting spam with genetic regular expressions. *SANS Institute InfoSec Reading Room*, 2007.

[4] R. Cox. Regular expression matching can be simple and fast (but is slow in java, perl, php, python, ruby, ...). `http://swtch.com/~rsc/regexp/regexp1.html`, January 2007.

[5] Hamish Cunningham, Diana Maynard, Kalina Bontcheva, Valentin Tablan, Niraj Aswani, Ian Roberts, Genevieve Gorrell, Adam Funk, Angus Roberts, Danica Damljanovic, Thomas Heitz, Mark A. Greenwood, Horacio Saggion, Johann Petrak, Yaoyong Li, and Wim Peters. *Text Processing with GATE (Version 6)*. 2011.

[6] Tomaž Erjavec. Multext-east: morphosyntactic resources for central and eastern european languages. *Language resources and evaluation*, 46(1):131–142, 2012.

[7] Information technology – Portable Operating System Interface (POSIX®) Base Specifications, Issue 7. ISO/IEC/IEEE 9945:2009, September 2009.

[8] Christopher Manning, Tim Grow, Teg Grenager, Jenny Finkel, and John Bauer. Stanford tokenizer. `http://nlp.stanford.edu/software/tokenizer.shtml`, 2010.

[9] G.A. Miller. WordNet: a lexical database for English. *Communications of the ACM*, 38(11):39–41, 1995.

[10] Diego Mollá, Menno Van Zaanen, and Daniel Smith. Named entity recognition for question answering. *Proceedings of ALTW*, pages 51–58, 2006.

[11] Christopher Potts. A Twitter-aware Tokenizer from the Sentiment Symposium Tutorial. `http://sentiment.christopherpotts.net/`, November 2011.

[12] B. Santorini. Part-of-speech tagging guidelines for the penn treebank project (3rd revision). Technical report, University of Pennsylvania, 1990.

[13] Helmut Schmid. Improvements in part-of-speech tagging with an application to german. In *In Proceedings of the ACL SIGDAT-Workshop*. Citeseer, 1995.

[14] Ovidiu Serban and Alexandre Pauchet. Agentslang: A fast and reliable platform for distributed interactive systems. In *Intelligent Computer Communication and Processing (ICCP), 2013 IEEE International Conference on*, pages 35–42. IEEE, 2013.

[15] Achim Stein. French TreeTagger Part-of-Speech Tags. `http://www.cis.uni-muenchen.de/~schmid/tools/TreeTagger/data/french-tagset.html`, April 2003.

[16] C. Kleene Stephen. Representation of events in nerve nets and finite automata. *Automata Studies*, pages 3 – 41, 1956.

[17] Daniela Xhemali, Chris J Hinde, and Roger G Stone. Genetic evolution of regular expressions for the automated extraction of course names from the web. In *GEM*, pages 118–124, 2010.

LITIS Laboratory, INSA de Rouen, Avenue de l'Université – BP 8, Saint-Étienne-du-Rouvray Cedex, 76801 France

E-mail address: `ovidiu.serban@insa-rouen.fr`

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LIX**, Number 1, 2014

A STUDY ON DYNAMIC CLUSTERING OF GENE EXPRESSION DATA

ADELA-MARIA SÎRBU

Abstract. Microarray and next-generation sequencing technologies allow measuring the levels of expressions of thousands of genes simultaneously. One of the most popular procedures used to analyze gene expression data is clustering. To study biological processes which evolve over time, researchers can either perform re-clustering from scratch every time new gene expression levels are available, which would be very time consuming, or adapt the previously obtained partitions using a dynamic clustering algorithm. This paper aims to investigate a couple of heuristics for centroids identification within a dynamic k-means based clustering algorithm that was previously introduced for clustering of gene expression data. Computational experiments on a real-life gene expression data set are provided, as well as an analysis of the obtained results.

1. Introduction

The emergence of microarray and next-generation sequencing technologies that allow measuring the levels of expressions of thousands of genes has lead to an exponential increase of the amount of gene expression data. In order to extract useful biological information from this data, exploratory analyses are performed. A first step in these analyses is clustering.

Clustering refers to creating a set of groups (clusters) and assigning each instance of a data set to one of these groups, according to a certain similarity measure. From a biological perspective, clustering represents an important step in determining gene functions, assuming that genes having similar expression levels under the same conditions may also have similar functions.

Received by the editors: December 8, 2013.

2010 *Mathematics Subject Classification.* 68P15, 68T05.

1998 *CR Categories and Descriptors.* I.2.6 [**Computing Methodologies**]: Artificial Intelligence – *Learning*; I.2.8 [**Computing Methodologies**]: Problem Solving, Control Methods, and Search – *Heuristic methods*.

Key words and phrases. Bioinformatics, Dynamic clustering.

We have previously introduced in [1] a novel approach for solving the dynamic problem of clustering gene expression data, when new features (expression levels for new points in time) are added to the genes within a data set. In this paper we aim to study two heuristics for centroids identification within the Core Based Dynamic Clustering of Gene Expression (CBDCGE) algorithm [1] and to analyze the influence on the initial centroids on the obtained results. The evaluations are performed on a data set that was used in [4].

The rest of the paper is organized as follows. Section 2 introduces the problem of dynamic gene expression clustering and presents an overview of the CBDCGE algorithm, together with two heuristics for centroids identification. A comparative study of the results, including experimental evaluations and analysis, is presented in Section 3. Section 4 outlines our conclusions and further work.

2. Background

2.1. **Dynamic Gene Clustering.** Gene expression data analysis is important within biology and medicine as the degree in which genes are expressed in different types of cell dictates cellular morphology and function. One of the most widely used data mining techniques used for this analysis is clustering.

Biological processes are mostly dynamic and in order to study and model them, scientists usually need information about gene expression at different moments in time, as the processes evolve. The resulting data sets are called time series data sets and they consist of gene expression data characterizing samples of cells or tissues which are extracted from the same individual at different moments in time, during the progression of the biological process. Thus, each gene is measured at several distinct time points and its expression levels are recorded. In the end, the time series data set consists of thousands of targeted genes (instances), each one being identified by a set of attributes (features): the values of its expression (quantified as real numbers) at all the considered time points.

The dynamism of gene expression data can be regarded from different perspectives: when new genes (instances) are added into the data set and when new gene expression levels (features), for the existing genes, are added into the data set. While for the first perspective there are several approaches in literature like k-means algorithms [9], artificial neural networks [10], particle swarm optimisation [11], for the second one, to our best knowledge, there are only two models that were previously introduced in [1] and [3].

Some biological processes only last for a short time, but there are other processes that may take months, even years (e.g. diseases). For the latter

ones, waiting until the process is finished to aquire all the necessary data is not feasible. An option would be to collect the data as the process evolves and apply the clustering algorithm each time new information is added. However, this technique could often be slow and inefficient, especially as the data sets contain thousands of instances and this increases the running time of the algorithm.

In order to surpass this drawback, a dynamic approach of clustering gene expression data has been introduced in [1], which is capable of adapting the previously obtained partition instead of re-clustering from scratch when new expression levels are added into the data set.

2.2. **Core Based Dynamic Clustering of Gene Expression.** In this section we present an overview of the CBDCGE model that was previously introduced in [1].

Let us denote by $\mathcal{G} = \{G_1, G_2, \ldots, G_n\}$ the set of genes to be classified. Each gene is measured at m moments in time and is therefore described by an m-dimensional vector $G_i = (G_{i1}, G_{i2}, \ldots, G_{im}), G_{ik} \in \Re, 1 \leq i \leq n, 1 \leq k \leq m$. An element G_{ik} from the vector characterizing the gene G_i represents the expression level of gene G_i at time point k.

Let $\{K_1, K_2, \ldots, K_p\}$ be the set of clusters discovered in data by applying the *k-means* algorithm. Each cluster is a set of genes, $K_j = \{G_{i_1}, G_{i_2}, \ldots, G_{i_{l_j}}\}$, $1 \leq l_j \leq n, 1 \leq i_k \leq n \; \forall 1 \leq k \leq l_j$, where l_j is the number of genes from cluster j. The centroid (cluster mean) of the cluster K_j is denoted by f_j, where $f_j = \left(\frac{\sum_{k=1}^{l_j} G_{i_k 1}}{l_j}, \ldots, \frac{\sum_{k=1}^{l_j} G_{i_k m}}{l_j} \right)$.

In order to measure the distance between genes, we have chosen the *Euclidian distance*, because it takes into account the magnitude of the changes in gene expression, therefore preserving more data [5].

The measured set of attributes consisting of m gene expression levels (coming from m consequent measurements) is afterwards extended with s ($s \geq 1$) new attributes, numbered as $(m+1), (m+2), \ldots, (m+s)$. After extension, the genes' feature vectors become $G'_i = (G_{i1}, \ldots, G_{im}, G_{i,m+1}, ...G_{i,m+s})$, $1 \leq i \leq n$.

We analyzed in [1] the problem of recalculating the genes' grouping into clusters, after gene extension and starting from the current partitioning. Our goal was to obtain a better performance with respect to the partitioning from scratch process.

We denoted by $K'_j, 1 \leq j \leq p$, the set containing the same genes as K_j, after the attribute set extension. By $f'_j, 1 \leq j \leq p$, we denote the mean (center) of the set K'_j.

The sets $K'_j, 1 \leq j \leq p$, will not necessarily represent clusters after the attribute set extension, as the newly arrived attributes can change the genes' arrangement into clusters. But there is a considerable chance, when adding one or few attributes to genes and when the attributes have equal weights and normal data distribution, that the old arrangement into clusters is close to the new actual one.

The actual clusters could be obtained by applying the *k-means* clustering algorithm on the set of extended genes, but this process is computationally expensive. That is why we tried to avoid this process and replace it with one less expensive but not less accurate. CBDCGE algorithm [1] starts from the partitioning obtained before the attribute set extension and adapts it considering the newly added gene expression levels. This way, the clustering of genes at intermediate time points during the experiment can be more efficiently exploited and the final result could be achieved in smaller amounts of time. More details about the CBDCGE algorithm and its characteristics may be found in [1].

For identifying the most appropriate number p of clusters in the gene expression data set, the following heuristic is used. We have determined p representative genes, i.e., a representative gene for each cluster. First, the initial number p of clusters is set to 0. Then the first representative gene is chosen as being the most "distant" gene from the set of all genes (the gene that maximizes the average distance from all other genes). The number p of chosen representatives becomes now 1. In order to choose the next representative gene we reason as follows: for each remaining gene (that was not already chosen), we compute the average distance (*davg*) from the gene and the already chosen representative genes. The next representative gene is chosen as the gene g that maximizes *davg* and this distance is greater than a positive given threshold (*distMin*), p is increased, and another representative gene is chosen again (the iterative process is performed again). If such a gene does not exist, it means that g is very close to all the already chosen representatives and should not be chosen as a new representative. In this case, the iterative process of selecting the initial centroids stops.

3. Comparative study

In this section we aim at providing an analysis of CBDCGE algorithm developed for dynamic clustering of gene expression data. The case study used

in our experiment, the evaluation measures, as well as the obtained results are presented and analysed in the following.

3.1. **Comparison criteria.** It is well known that a problem of the k-means based clustering algorithms is that they are sensitive to the selection of the initial centroids and may converge to a local minimum of the squared error value if the initial centroids are not properly chosen [12]. Consequently, it is very likely that the initial centroids may have an impact on the accuracy of the obtained results.

Thus, our comparative analysis is oriented to different methods for centroids' identification within the CBDCGE algorithm, as follows:

Heuristic 1. Te first heuristic method for selecting centroids is the one used in [1].

Heuristic 2. The second heuristic method for selecting the appropriate number p of clusters is based on selecting p representatives genes, as follows.

(i) The initial number p of clusters is set to 0.
(ii) The first representative gene chosen is the most "distant" gene from the set of all genes (the gene that maximizes the average distance from all other genes). The number p of chosen representatives becomes 1.
(iii) In order to choose the next representative gene we reason as follows. For each remaining gene (that was not already chosen), we compute the minimum distance (*dmin*) from the gene and the already chosen representative genes. The next representative gene is chosen as the gene g that maximizes *dmin* and this distance is greater than a positive given threshold (*distMin*), p is increased, and step (iii) is performed again. If such a gene does not exist, it means that g is very close to all the already chosen representatives and should not be chosen as a new representative. In this case, the iterative process of selecting the initial centroids stops.

Random. The third way of choosing centroids is a random selection of p centroids, p being the number of clusters heuristically identified as above.

3.2. **Experiments.** In order to test the performance of the CBDCGE algorithm, we used a real-life data set, taken from [4] which contains the levels of expression of 6400 genes belonging to organism *Saccharomyces cerevisiae* during its metabolic shift from fermentation to respiration.

We have chosen this dataset from the following reasons: it is time series gene expression dataset, it is publicly available, it was used in several approaches from the literature giving us the posibility to compare our results

with the existing ones, and allows us to perform an evaluation from a biological perspective.

Gene expression levels were measured at seven time points during the diauxic shift. First, a pre-processing step was applied, in which the genes having small variance over time or very low absolute expression values, as well as genes whose profiles have low entropy are removed .

Considering an initial number of features (denoted by m) characterizing the genes from the considered data set, the experiments are conducted as follows:

(1) The number of clusters nc and the initial centroids are identified in the data set using different selection criteria (Subsection 3.1). The *k-means* clustering algorithm is applied on the data set consisting of m-dimensional genes, starting from the identified centroids and a partition $\mathcal{K}$ is provided.
(2) The set of features is now extended with s ($s \geq 1$) new attributes, numbered as $(m+1), (m+2), \ldots, (m+s)$. The $CBDCGE$ adaptive algorithm is now applied, by adapting the partition $\mathcal{K}$ and considering the instances extended with the newly added s features.
(3) The partition into clusters provided by $CBDCGE$ algorithm (denoted by $\mathcal{K}_{CBDCGE}$) is compared with the one provided by the $k-means$ algorithm applied from scratch on the $m+s$-dimensional instances (denoted by $\mathcal{K}'$). We mention that the initial centroids considered in the partitional clustering process are the centroids identified at step 1. The comparison of the obtained partitions is made considering the evaluation measures presented in Subsection 3.2.1 (both from the clustering and biological point of view), as well as the number of iterations performed by the clustering algorithms.

In order to accurately evaluate CBDCGE algorithm, we considered the same initial centroids when running k-means for the initial and feature-extended gene set (m and m + s number of features).

3.2.1. *Evaluation measures.* In order to measure the quality of the obtained partitions we use four *evaluation measures*. The first three measures (*IntraD*, *Dunn* and *Dist*) evaluate a partition from the clustering point of view, while the last one (*Z-score*) evaluates a partition from a biological point of view.

In the following, let us consider a partition $K = \{K_1, \ldots, K_p\}$, where each cluster consists of a set of genes. In the following $d(G_i, G_j)$ denotes the euclidean distance between G_i and G_j.

A. Intra-cluster distance of a partition - IntraD. The *intra-cluster distance* of a partition K, denoted by *IntraD(K)*, is defined as:

$$IntraD(K) = \sum_{j=1}^{p} \sum_{k=1}^{l_j} d(G_{i_k}, f_j)$$

where the cluster K_j is a set of genes $\{G_{i_1}, G_{i_2}, \ldots, G_{i_{l_j}}\}$ and f_j is the centroid (mean) of K_j.

From the point of view of a clustering technique, smaller values for $IntraD$ indicate better partitions, meaning that $IntraD$ has to be minimized.

B. Dunn Index - Dunn. The *Dunn index* [6] of a partition K is defined as:

$$Dunn(K) = \frac{d_{min}}{d_{max}}$$

where d_{min} represents the smallest distance between two genes from different clusters and d_{max} is the largest distance among two genes from the same cluster. The *Dunn index* takes values from the interval [0,∞]. The greater the value of this index, the better a partition is, therefore the *Dunn index* should be maximized.

C. Overall distance of a partition - Dist [1]. The *overall distance* of a partition K, denoted by *Dist(K)*, is defined as:

$$Dist(K) = \sum_{j=1}^{p} d_j$$

where d_j is defined as the sum of distances between all pair of genes from the cluster K_j, i.e

$$d_j = \sum_{G_1, G_2 \in K_j} d(G_1, G_2)$$

From the point of view of a clustering technique, smaller values for $Dist$ indicate better partitions, meaning that $Dist$ has to be minimized.

D. Z-score. Z-score [7] is a figure of merit, indicating the relationship between a clustering result and the functional annotation of the used genes, within the gene ontology developed by the Gene Ontology Consortium [8]. A higher value of the z-score indicates that the obtained clusters are more biologically relevant and therefore a more accurate clustering. To compute the z-score for a partition we used the ClusterJudge software, which implements the algorithm described in [7].

No.	**No. of clusters**	**No.of iterations**	**IntraD**	**Dunn**	**Dist**	**Z-score**
1	Heuristc 1 $distMin = 3.47$ $nc = 44$					
$\mathcal{K}'$	44	21	448.1514	0.1586	392.0747	7.7170
$\mathcal{K}_{CBDCGE}$	40	14	445.0609	0.1894	412.4337	9.9510
2	Heuristc 2 $distAvg = 1.13$ $nc = 44$					
$\mathcal{K}'$	44	11	440.1101	0.2686	20685.6718	6.0420
$\mathcal{K}_{CBDCGE}$	41	12	448.1529	0.2102	17133.5209	7.6540
3	Random $nc = 44$					
$\mathcal{K}'$	44	15	424.8114	0.1363	10912.5659	7.8844
$\mathcal{K}_{CBDCGE}$	43	14	427.7379	0.1535	11593.0304	9.2200

TABLE 1. Results for the first experiment.

3.3. **Results and Discussion.** In order to decide the most appropriate heuristic for selecting the initial centroids in the adaptive clustering process, we conducted two experiments. In each one we started from a different number of initial features and then we added the rest of the attributes (up to seven, which is the total number of attributes).

In both experiments the centroids were identified in three ways: using *Heuristic 1*, *Heuristic 2* and randomly (Section 3.1). For the randomly chosen centroids, an average obtained by five consequent runs was provided.

3.3.1. *Experiment 1.* In this experiment, the initial data set contains the first five features ($m = 5$) features and the remaining two features ($s = 2$) are added subsequently. The obtained results are presented in Table 1 and Figures 1a-3a.

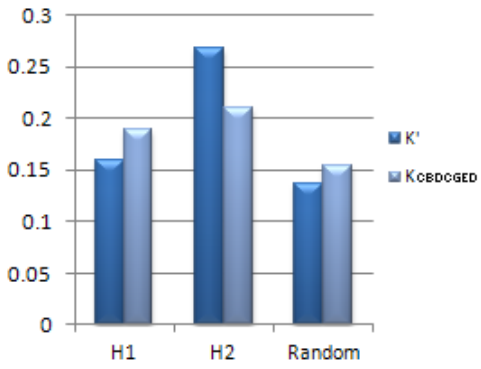
From these results we can conclude the following:

- The minimum number of iterations and the smallest $Dist$ value are acheived by using Heuristic 1, both in K' and K_{CBDCGE}.
- The smallest $IntraD$ value is achieved by randomly choosing centroids, both in K' and K_{CBDCGE}.
- The highest $Dunn$ value is achieved by using Heuristic 2, both in K' and K_{CBDCGE}.
- The highest $Z-score$ value is achieved by randomly choosing centroids in K' and using Heuristic 1 in K_{CBDCGE}.
- From a biological point of view (considering the $Z - score$ evaluation measure), in all three cases the adaptive clustering outperforms the re-clustering from scratch process.

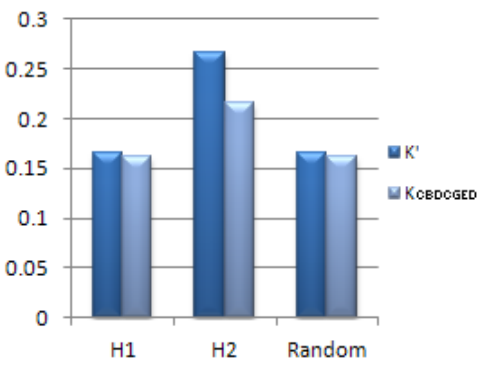
No.	**No. of clusters**	**No.of iterations**	**IntraD**	**Dunn**	**Dist**	**Z-score**
1	Heuristc 1 $distMin = 4.66$ $nc = 42$					
$\mathcal{K}'$	42	23	451.5669	0.1655	446.6040	8.2300
$\mathcal{K}_{CBDCGE}$	40	23	438.2164	0.1621	351.8567	8.5044
2	Heuristc 2 $distAvg = 1.51$ $nc = 42$					
$\mathcal{K}'$	42	18	445.1501	0.2664	21649.7054	7.288
$\mathcal{K}_{CBDCGE}$	40	18	436.7869	0.2163	16133.5829	9.244
3	Random $nc = 42$					
$\mathcal{K}'$	42	15	430.6806	0.1434	11368.6995	9.0144
$\mathcal{K}_{CBDCGE}$	41	14	428.3622	0.1949	11848.0839	9.0853

TABLE 2. Results for the second experiment.

3.3.2. *Experiment 2.* In this experiment, the initial data set contains the first six features ($m = 6$) features and the remaining feature ($s = 1$) is added afterwards. The obtained results are presented in Table 2 and Figures 1b-3b.



(A) First experiment.

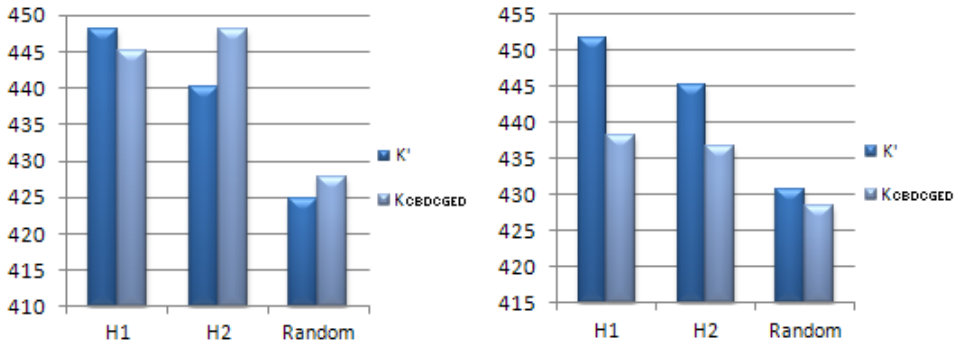


(B) Second experiment.

FIGURE 1. Illustration of the values of Dunn index obtained by using Heuristic 1, Heuristic 2 and random centroids, both for K' and K_{CBDCGE}.

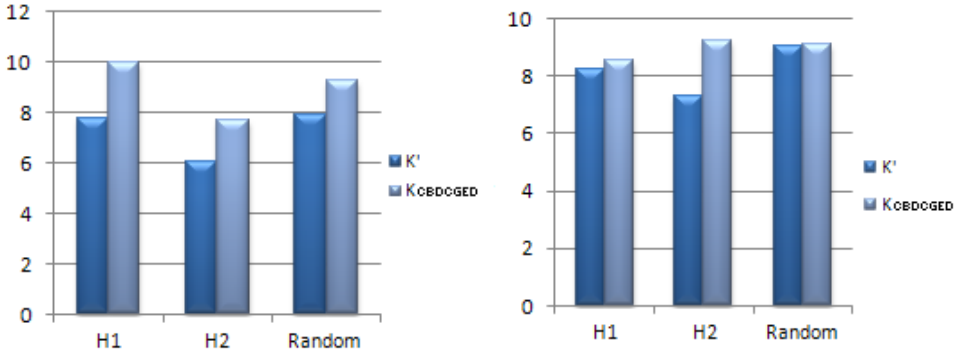
From these results we can conclude the following:

- The minimum number of iterations and the smallest $IntraD$ value are acheived by randomly choosing centroids, both in K' and K_{CBDCGE}.
- The highest $Dunn$ value is achieved by using Heuristic 2, both in K' and K_{CBDCGE}.



(A) First experiment. (B) Second experiment.

FIGURE 2. Illustration of the values of IntraD obtained by using Heuristic 1, Heuristic 2 and random centroids, both for K' and K_{CBDCGE}.



(A) First experiment. (B) Second experiment.

FIGURE 3. Illustration of the values of Z-score obtained by using Heuristic 1, Heuristic 2 and random centroids, both for K' and K_{CBDCGE}.

- The smallest $Dist$ value is achieved by using Heuristic 1, both in K' and K_{CBDCGE}.
- The highest $Z-score$ value is achieved by randomly choosing centroids in K' and using Heuristic 2 in K_{CBDCGE}.
- The $Z-score$ evaluation measure, indicates in all three cases that the adaptive clustering outperforms the re-clustering from scratch.

3.3.3. *Statistical analysis.* Since for the problem we approach in this paper, clustering of gene expression data, the most relevant evaluation measure is the biological one, we performed a statistical analysis of Z-score values. We computed 95% Confidence Interval [2] for the average of the differences between the Z-scores obtained using the adaptive and from scratch approaches. All the Z-score values from the two experiments were considered. We obtained the (0.53, 1.95) Confidence Interval for the average. Thus, there is a 95% confidence that the Z-score of the partition obtained adaptively exceeds the Z-score of the partition obtained by applying the k-means from scratch with a value that lies within the specified range.

Due to the variation of the results, we can not conclude which heuristic is the best. It depends on the evaluation measure (e.g. Heuristic 2 is the best from *Dunn* index perspective, but is not the best from $IntraD$ perspective), the type of algorithm (adaptive/from scratch), the number of features added (for the adaptive approach). Even if there are cases in which choosing centroids randomly gives better results than using heuristics, it does not represent a reliable option, as an inappropriate choice could strongly degenerate results.

Still, for both experiments we have performed, we can conclude that from a biological point of view (considering the $Z - score$ evaluation measure) a better approach is to use an heuristic for the initial centroids selection, instead of a random choice.

4. Conclusions and further work

In this paper we presented a study on CBDCGE algorithm for dynamic clustering of gene expression data, with focus on the impact of heuristics used for centroids identification on the quality of clustering.

After an analysis of the obtained results, we can conclude, from a biological perspective, that CBDCGE algorithm outperforms the k-means applied from scratch, as it obtained better Z-score values in all the investigated cases.

As further work we plan to extend the CBDCGE method to a fuzzy clustering approach. Moreover, we plan to examine practical applications of the proposed method and to extend the experimental evaluation on other publicly available case studies.

References

[1] Bocicor, Maria Iuliana and Sirbu, Adela and Czibula, Gabriela *Dynamic core based clustering of gene expression data*, International Journal of Innovative Computing, Information and Control, volume 10, no. 3, P.1-13, 2014

[2] Brown, LD, Cat, TT and DasGupta, A *Interval Estimation for a proportion*, Statistical Science, volume 16, P.101-133, 2001

[3] Sirbu, Adela and Bocicor, Maria Iuliana *A Dynamic Approach for Hierarchical Clustering of Gene Expression Data*, Proceedings of 9th International Conference On Intelligent Computer Communication and Processing, P.3-6, 2013

[4] DeRisi, J.L. and Iyer, P.O. and Brown, V.R. *Exploring the Metabolic and Genetic Control of Gene Expression on a Genomic Scale*, International Journal of Innovative Computing, Information and Control, volume 278, no. 5338, P.680-686, 1997

[5] Kim, K. and Zhang, S. and Jiang, K. and Cai, L. and Lee, I.B. and Feldman, L.J. and Huang, H. *Measuring similarities between gene expression profiles through new data transformations*, BMC Bioinformatics, volume 8, no. 29, 2007

[6] M. K. Pakhira and S. Bandyopadhyay and U. Maulik *Validity index for crisp and fuzzy clusters*, Pattern Recognition, volume 37, no. 3, P.478-501, 2004

[7] Gibbons, F.D. and Roth, F.P. *Judging the Quality of Gene Expression-Based Clustering Methods Using Gene Annotation*, Genome Research, volume 12, no. 10, P.1574-1581, 2002

[8] Ashburner, M. and Ball, C.A. and Blake, J.A. and Botstein, D. and Butler, H. and Cherry, J.M. and Davis, A.P. and Dolinski, K. and Dwight, S.S. and Eppig, J.T. and et al. *Gene ontology: tool for the unification of biology. The Gene Ontology Consortium*, Nature Genetics, volume 25, no. 1, P.25-29, 2000

[9] Bagirov, A.M. and Mardaneh, K. *Modified global k-means algorithm for clustering in gene expression data sets*, Proceedings of the 2006 workshop on Intelligent systems for bioinformatics, series WISB '06, P.23-28, 2006

[10] Yuhui, Y. and Lihui, C. and Goh, A. and Wong, A. *Clustering gene data via associative clustering neural network*, Proc. 9th Intl. Conf. on Information Processing, P.2228-2232, 2002

[11] Xiao, X. and Dow, E.R. and Eberhart, R.C. and Miled, Z.B. and Oppelt, R.J. *Gene Clustering Using Self-Organizing Maps and Particle Swarm Optimization*, Proc. 17th Intl. Symposium on Parallel and Distributed Processing, 2003

[12] Mohammed El Agha, Wesam M. Ashour. *Efficient and Fast Initialization Algorithm forK-means Clustering*, MECS Publisher, no. 1, 2012

Babeş-Bolyai University, Department of Computer Science, 1 M. Kogălniceanu Street, 400084 Cluj-Napoca, Romania

E-mail address: adela@cs.ubbcluj.ro

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LIX**, Number 1, 2014

AN INFEASIBLE FULL-NEWTON STEP ALGORITHM FOR LINEAR OPTIMIZATION WITH ONE CENTERING STEP IN MAJOR ITERATION

ZSOLT DARVAY, INGRID-MAGDOLNA PAPP, AND PETRA-RENÁTA TAKÁCS

Abstract. Recently, Roos proposed a full-Newton step infeasible interior-point method (IIPM) for solving linear optimization (LO) problems. Later on, more variants of this algorithm were published. However, each main step of these methods is composed of one feasibility step and several centering steps. The purpose of this paper is to prove that by using a new search direction it is enough to take only one centering step in order to obtain a polynomial-time method. This algorithm has the same complexity as the best known IIPMs.

1. Introduction

In this paper, we define a new interior-point algorithm (IPA) for LO, which approximates the optimal solution, starting from infeasible points. Karmarkar's publication [7] appeared in 1984 and meant a paradigm shift in the area of optimization algorithms. Following this, a large amount of IPAs has been published. These algorithms have many applications in different fields, such as engineering, economics, transportation, statistics, machine learning and data mining. The first infeasible methods were developed by Lustig [9] and Tanabe [18]. The complexity of IPAs was analysed at first by Kojima, Meggido, Mizuno [8] and Zhang [23]. Bonnans and Potra [3] defined infeasible algorithms for linear complementarity problems. The predictor-corrector method for LO problem was studied by Potra [14, 15]. We can read about new results on infeasible interior-point algorithms in books wrote by Wright [21], Ye [22] and Vanderbei [19].

Received by the editors: March 1, 2014.

2010 *Mathematics Subject Classification.* 90C05, 90C51.

1998 *CR Categories and Descriptors.* G.1.6. [**Mathematics of Computing**]: Numerical Analysis – *Optimization – Linear programming*.

Key words and phrases. Linear optimization, Infeasible interior-point method, Newton's method, Polynomial complexity.

Roos [16] introduced a new algorithm, which uses only full-Newton steps and starts from infeasible points. Mansouri and Roos [10] proposed a simplified IIPM. Gu et al. [6] presented an improved variant of the algorithm. Darvay [4, 5] defined a new technique for finding search directions for LO problems. Achache [1] generalized this approach to convex quadratic optimization, and Wang and Bai [20] to symmetric optimization. Ahmadi, Hasani and Kheirfam [2] adapted this technique to IIPMs. Pan, Li and He [13] proposed an IIPM using a logarithmic equivalent transformation of the centering equations. Mansouri, Siyavash and Zangiabadi [11] extended the algorithm introduced by Roos to semidefinite optimization problems using the method proposed in [5].

In the full-Newton step IIPMs defined in these papers two types of steps are used, one feasibility step and a few centering steps. In this paper, we present a new full-Newton step IIPM based on the technique introduced in [4, 5] and we prove that it suffices to take only one centering step in order to get a well-defined algorithm.

We introduce some notations used throughout the paper. Let x and s be two n-dimensional vectors. Then, xs denotes the componentwise product of the vectors x and s. Similarly, we define $\frac{x}{s} = \left[\frac{x_1}{s_1}, \frac{x_2}{s_2}, ..., \frac{x_n}{s_n}\right]^T$, where $s_i \neq 0$ for all $1 \leq i \leq n$. If $x \geq 0$, then $\sqrt{x}$ is the vector obtained by taking square roots of the components of x. Let e be the n-dimensional all-one vector. Furthermore, $diag(x)$ is a diagonal matrix, which contains on his main diagonal the elements of x in the original order. Besides these, $\|x\|$ denotes the Euclidean norm, $\|x\|_\infty$ the Chebyshev norm, $\|x\|_1$ the 1-norm, and $min(x)$ the minimal component of x. Finally, if $f(t) \geq 0$ and $g(t) \geq 0$ are real valued functions, then $f(t) = O(g(t))$ means that there exists a positive constant γ so that $f(t) \leq \gamma g(t)$.

The paper is organized in the following way. Firstly, we present the LO problem. In the next section the feasible primal-dual algorithm and its complexity analysis are revisited. Then, we introduce the perturbed problems and the new infeasible primal-dual algorithm. The purpose of the next sections is to provide the complexity analysis of the algorithm and to prove its polynomiality. Finally, the paper ends up with a conclusion.

2. The Linear Optimization Problem

Let us consider the following primal problem

$$(P) \qquad \begin{aligned} &min\ c^T x, \\ &Ax = b, \\ &x \geq 0, \end{aligned}$$

where $A \in \mathbb{R}^{m\times n}$, $rank(A) = m$, $b \in \mathbb{R}^m$ and $c \in \mathbb{R}^n$.
The dual of this problem is

$$(D) \qquad \begin{aligned} &max\ b^T y, \\ &A^T y + s = c, \\ &s \geq 0. \end{aligned}$$

In case of the feasible LO algorithms we assume that the *interior-point condition* (IPC) holds for the primal and dual problems, i.e., there exists (x^0, y^0, s^0) so that

$$(IPC) \qquad \begin{aligned} Ax^0 &= b, \qquad x^0 > 0, \\ A^T y^0 + s^0 &= c, \qquad s^0 > 0. \end{aligned}$$

Using the self-dual embedding technique we can always construct a LO problem in such a way that the IPC holds. So, the IPC can be assumed without loss of generality. Furthermore, the self-dual embedding model yields $x^0 = s^0 = e$. Denote $\mu^0 = \frac{(x^0)^T s^0}{n} = 1$.
The optimal solution of the primal-dual pair is characterized by the following system of equations:

$$(1) \qquad \begin{aligned} Ax &= b, \qquad x \geq 0, \\ A^T y + s &= c, \qquad s \geq 0, \\ xs &= 0. \end{aligned}$$

The first and the second equation of system (1) are called *feasibility conditions*. They serve for maintaining feasibility. The last equation is named *complementarity condition*. Primal-dual interior-point methods replace the complementarity condition with a parameterized equation. Hence we obtain:

$$(2) \qquad \begin{aligned} Ax &= b, \qquad x \geq 0, \\ A^T y + s &= c, \qquad s \geq 0, \\ xs &= \mu e, \end{aligned}$$

where $\mu > 0$. If the IPC holds, then for a fixed $\mu > 0$ the system (2) has a unique solution, called the μ-*center* or *analytic center* (Sonnevend [17]). The set of μ-centers for $\mu > 0$ formes a well-behaved curve, called *central path*. As μ tends to zero, the central path converges to the optimal solutions of (P) and (D).

3. Feasible Primal-Dual Algorithm

In this section we present the new technique for finding search directions introduced in [5]. Let $\mathbb{R}^+ = \{x \in \mathbb{R} \mid x \geq 0\}$ and $\varphi : \mathbb{R}^+ \to \mathbb{R}^+$ be a continuously

differentiable and invertible function. The system of equations, which defines the central path (2) can be written in the following equivalent form:

$$\begin{aligned} Ax &= b, \qquad x \geq 0, \\ A^T y + s &= c, \qquad s \geq 0, \\ \varphi\left(\frac{x_i s_i}{\mu}\right) &= \varphi(1), \text{ for all } 1 \leq i \leq n. \end{aligned} \tag{3}$$

Applying Newton's method to (3) we can obtain new search directions. If $\varphi(t) = \sqrt{t}$, then we get the following system:

$$\begin{aligned} A\Delta x &= 0, \\ A^T \Delta y + \Delta s &= 0, \\ s\Delta x + x\Delta s &= 2(\sqrt{\mu x s} - xs). \end{aligned} \tag{4}$$

We can give a proximity measure to the central path [5]:

$$\sigma(xs, \mu) = \left\| e - \sqrt{\frac{xs}{\mu}} \right\|. \tag{5}$$

The feasible primal-dual algorithm can be described as in Figure 1.

Feasible primal-dual algorithm [5]

Let $\epsilon > 0$ be the accuracy parameter, $0 < \theta < 1$ the update parameter (default $\theta = \frac{1}{2\sqrt{n}}$) and $0 < \tau < 1$ the proximity parameter (default $\tau = \frac{1}{2}$). Assume that for (x^0, y^0, s^0) the IPC holds, and $\mu^0 = \frac{(x^0)^T s^0}{n}$. Furthermore, suppose that $\sigma(x^0 s^0, \mu^0) < \tau$.

begin
  $(x, y, s) := (x^0, y^0, s^0)$;
  $\mu := \mu^0$;
  while $x^T s > \epsilon$ **do begin**
    $\mu := (1 - \theta)\mu$;
    calculate $(\Delta x, \Delta y, \Delta s)$ *from (4)*
    $x := x + \Delta x$;
    $y := y + \Delta y$;
    $s := s + \Delta s$;
  end
end.

FIGURE 1. Feasible primal-dual algorithm

The following lemmas (cf. [5]) are meant to prove the polynomiality of the algorithm. Let $x_+ = x + \Delta x$ and $s_+ = s + \Delta s$ be the vectors we get after a full-Newton step.

Lemma 3.1 *Let $\sigma = \sigma(xs, \mu) < 1$. Then $x_+ > 0$ and $s_+ > 0$, so the full-Newton step is strictly feasible.*

Lemma 3.2 *Let $\sigma = \sigma(xs, \mu) < 1$. Then*

$$\sigma(x_+ s_+, \mu) \leq \frac{\sigma^2}{1 + \sqrt{1 - \sigma^2}},$$

which means that the full-Newton step ensures local quadratic convergence of the proximity measure.

Lemma 3.3 *Let $\sigma = \sigma(xs, \mu)$. Then*

$$(x_+)^T s_+ = \mu(n - \sigma^2),$$

thus $(x_+)^T s_+ \leq \mu n$.

Lemma 3.4 *Let $\sigma = \sigma(xs, \mu) < 1$ and $\mu_+ = (1-\theta)\mu$, where $0 < \theta < 1$. Then*

$$\sigma(x_+ s_+, \mu_+) \leq \frac{\theta\sqrt{n} + \sigma^2}{1 - \theta + \sqrt{(1-\theta)(1-\sigma^2)}}.$$

Furthermore, if $\sigma < \frac{1}{2}$, $\theta = \frac{1}{2\sqrt{n}}$ and $n \geq 4$, then $\sigma(x_+ s_+, \mu_+) < \frac{1}{2}$.

Lemma 3.5 *Suppose that (x^0, s^0) are strictly feasible, $\mu^0 = \frac{(x^0)^T s^0}{n}$ and $\sigma(x^0 s^0, \mu^0) < \frac{1}{2}$. Let x^k and s^k be the vectors obtained after k iterations. Then, for every*

$$k \geq \left\lceil \frac{1}{\theta} \log \frac{(x^0)^T s^0}{\epsilon} \right\rceil$$

we get $(x^k)^T s^k \leq \epsilon$.

Lemma 3.6 *Assume that $x^0 = s^0 = e$. Then, Algorithm 1 demands no more than*

$$\left\lceil \frac{1}{\theta} \log \frac{n}{\epsilon} \right\rceil$$

interior-point iterations.

Theorem 3.7 *Suppose that $x^0 = s^0 = e$. Using the default values for θ and τ we get that Algorithm 1 requires at most*

$$O\left(\sqrt{n} \log \frac{n}{\epsilon}\right)$$

interior-point iterations. The resulting vectors satisfy $x^T s \leq \epsilon$.

4. The Perturbed Problems

From now on we don't assume that the initial points are feasible solutions of the primal and dual problems, but we suppose that an optimal solution exists. Let $\zeta > 0$ be given so that

$$\|\bar{x} + \bar{s}\|_\infty \leq \zeta, \tag{6}$$

where $\bar{x}$ and $(\bar{y}, \bar{s})$ are optimal solutions of (P) and (D). Hence, the algorithm will start with the following initial iterates:

$$x^0 = s^0 = \zeta e, \qquad y^0 = 0, \qquad \mu^0 = \zeta^2. \tag{7}$$

Instead of the original problem, we consider the following perturbed problem, which was studied by many researchers (see for example Ye [22] and Roos [16]):

$$(P_\nu) \qquad \begin{array}{c} min\ (c - \nu(c - A^T y^0 - s^0))^T x, \\ Ax = b - \nu(b - Ax^0), \\ x \geq 0, \end{array}$$

and its dual problem:

$$(D_\nu) \qquad \begin{array}{c} max\ (b - \nu(b - Ax^0))^T y, \\ A^T y + s = c - \nu(c - A^T y^0 - s^0), \\ s \geq 0, \end{array}$$

where $0 < \nu \leq 1$. The following lemma holds.

Lemma 4.1 (*cf. [22], Theorem 5.13). The problems (P) and (D), are feasible if and only if for each ν satisfying $0 < \nu \leq 1$ the perturbed problems (P_ν) and (D_ν) satisfy the (IPC).*

The system of equations, which defines the central path of the perturbed problems can be written in the following form:

$$\begin{array}{rcll} b - Ax & = & \nu(b - Ax^0), & x \geq 0, \\ c - A^T y - s & = & \nu(c - A^T y^0 - s^0), & s \geq 0, \\ xs & = & \mu e. & \end{array} \tag{8}$$

Let us consider the function φ defined in Section 3. Then the system (8) is equivalent to

$$\begin{array}{rcll} b - Ax & = & \nu(b - Ax^0), & x \geq 0, \\ c - A^T y - s & = & \nu(c - A^T y^0 - s^0), & s \geq 0, \\ \varphi\left(\dfrac{x_i s_i}{\mu}\right) & = & \varphi(1), \text{ for all } 1 \leq i \leq n. & \end{array} \tag{9}$$

Now we apply Newton's method for system (9). Assuming that $\varphi(t) = \sqrt{t}$, and x and (y, s) are strictly feasible solutions of (P_ν) and (D_ν), we obtain system (4).

5. A New Primal-Dual Algorithm

Let $\nu_+ = (1-\theta)\nu$, where $0 < \theta < 1$. Let us introduce the following notations:

$$r_b^0 = b - Ax^0, \quad r_c^0 = c - A^T y^0 - s^0.$$

Assuming that x and (y, s) are strictly feasible solutions of (P_ν) and (D_ν), we define the $(\Delta^f x, \Delta^f y, \Delta^f s)$ step in order to get feasible solutions of (P_{ν_+}) and (D_{ν_+}). Thus, using $\varphi(t) = \sqrt{t}$, we obtain the following system:

$$\begin{aligned} A\Delta^f x &= \theta\nu r_b^0, \\ A^T\Delta^f y + \Delta^f s &= \theta\nu r_c^0, \\ s\Delta^f x + x\Delta^f s &= 2(\sqrt{\mu xs} - xs). \end{aligned} \tag{10}$$

We introduce the following notations:

$$v = \sqrt{\frac{xs}{\mu}}, \qquad d_x = \frac{v\Delta^f x}{x}, \qquad d_s = \frac{v\Delta^f s}{s},$$

we obtain

$$\mu v(d_x + d_s) = s\Delta^f x + x\Delta^f s \tag{11}$$

and

$$d_x d_s = \frac{\Delta^f x \Delta^f s}{\mu}. \tag{12}$$

Using these notations we get the scaled form of system (10):

$$\begin{aligned} \bar{A}d_x &= \frac{\theta\nu}{\mu} r_b^0, \\ \bar{A}^T\Delta^f y + d_s &= \theta\nu \frac{r_c^0 v}{s}, \\ d_x + d_s &= p_v, \end{aligned} \tag{13}$$

where $p_v = 2(e - v)$ and $\bar{A} = \frac{1}{\mu} A diag\left(\frac{x}{v}\right)$. The proximity measure defined by (5) can be written as follows:

$$\sigma(v) = \sigma(xs, \mu) = \frac{\|p_v\|}{2} = \|e - v\|.$$

Let $q_v = d_x - d_s$. Then

$$d_x = \frac{p_v + q_v}{2}, \quad d_s = \frac{p_v - q_v}{2}.$$

Multiplying the two equalities, we get

$$\frac{q_v^2}{4} = \frac{p_v^2}{4} - d_x d_s. \tag{14}$$

It follows that

$$\frac{\|q_v\|^2}{4} = \frac{\|p_v\|^2}{4} - d_x^T d_s. \tag{15}$$

The algorithm is defined in Figure 2.

Infeasible primal-dual algorithm

Let $\epsilon > 0$ be the accuracy parameter and $0 < \theta < 1$ the update parameter (default $\theta = \frac{1}{8n}$). We assume that the initial points are (x^0, y^0, s^0), $x^0 > 0$, $s^0 > 0$ and $x^0 s^0 = \mu^0 e$ (default $x^0 = \zeta e$, $y^0 = 0$, $s^0 = \zeta e$, $\mu^0 = \zeta^2$, where $\zeta > 0$).

begin
  $(x, y, s) := (x^0, y^0, s^0)$;
  $\mu := \mu^0$; $\nu := 1$;
  while $max(x^T s, \|b - Ax\|, \|c - A^T y - s\|) \geq \epsilon$ **do begin**
    $(x, y, s) := (x, y, s) + (\Delta^f x, \Delta^f y, \Delta^f s)$;
    $\mu := (1 - \theta)\mu$;
    $\nu := (1 - \theta)\nu$;
    $(x, y, s) := (x, y, s) + (\Delta x, \Delta y, \Delta s)$;
  end
end.

FIGURE 2. Infeasible primal-dual algorithm

In the following sections we analyse the complexity of the algorithm.

6. Analysis of the Algorithm

Let $x^f = x + \Delta^f x$ and $s^f = s + \Delta^f s$ be the vectors obtained after the feasibility step. In the next lemma we give a condition, which guarantees the feasibility of x^f and s^f. Let $0 < \theta < 1$ and denote

$$\omega(v) = \frac{1}{2}\sqrt{\|d_x\|^2 + \|d_s\|^2},$$

$$v^f = \sqrt{\frac{x^f s^f}{\mu}}, \quad v^+ = \sqrt{\frac{x^f s^f}{\mu_+}}, \quad \mu_+ = (1 - \theta)\mu.$$

Lemma 6.1 *Let $x > 0$ be a feasible solution of (P_ν) and $s > 0$ a feasible solution of (D_ν), and $\sigma(v) = \sigma(xs, \mu)$, which satisfies $\sigma(v)^2 + 2\omega(v)^2 < 1$. Then we have*

$$x^f > 0 \quad \text{and} \quad s^f > 0, \tag{16}$$

thus x^f and s^f are strictly feasible solutions of (P_{ν_+}) and (D_{ν_+}).

Proof. From the definition of x^f and s^f and system (10) we deduce that we have to prove (16). For each $0 \le \alpha \le 1$ denote $x^f(\alpha) = x + \alpha\Delta^f x$ and $s^f(\alpha) = s + \alpha\Delta^f s$. Thus

$$x^f(\alpha)s^f(\alpha) = xs + \alpha(s\Delta^f x + x\Delta^f s) + \alpha^2\Delta^f x\Delta^f s.$$

Using (11) and (12) we may write

$$\frac{1}{\mu}x^f(\alpha)s^f(\alpha) = v^2 + \alpha v(d_x + d_s) + \alpha^2 d_x d_s. \tag{17}$$

By (14) we have

$$\frac{1}{\mu}x^f(\alpha)s^f(\alpha) = (1-\alpha)v^2 + \alpha(v^2 + vp_v) + \alpha^2\left(\frac{p_v^2}{4} - \frac{q_v^2}{4}\right).$$

Moreover, from $p_v = 2(e - v)$ we get

$$v^2 + vp_v = 2v - v^2 = e - (e-v)^2 = e - \frac{p_v^2}{4}, \tag{18}$$

so

$$\frac{1}{\mu}x^f(\alpha)s^f(\alpha) = (1-\alpha)v^2 + \alpha\left(e - (1-\alpha)\frac{p_v^2}{4} - \alpha\frac{q_v^2}{4}\right). \tag{19}$$

The inequality $x^f(\alpha)s^f(\alpha) > 0$ holds if

$$\left\|(1-\alpha)\frac{p_v^2}{4} + \alpha\frac{q_v^2}{4}\right\|_\infty < 1.$$

Using (15) we obtain

$$\left\|(1-\alpha)\frac{p_v^2}{4} + \alpha\frac{q_v^2}{4}\right\|_\infty \le (1-\alpha)\frac{\|p_v^2\|_\infty}{4} + \alpha\frac{\|q_v^2\|_\infty}{4} \le$$

$$\le (1-\alpha)\frac{\|p_v\|^2}{4} + \alpha\frac{\|q_v\|^2}{4} = \sigma(v)^2 - \alpha d_x^T d_s.$$

Moreover,

$$-d_x^T d_s \le |d_x^T d_s| \le \|d_x\|\,\|d_s\| \le \frac{1}{2}\left(\|d_x\|^2 + \|d_s\|^2\right) = 2\omega(v)^2. \tag{20}$$

Using this inequality we may write

$$\sigma(v)^2 - \alpha d_x^T d_s \le \sigma(v)^2 + 2\omega(v)^2 < 1,$$

thus we obtain that for each $0 \leq \alpha \leq 1$ *the* $x^f(\alpha)s^f(\alpha) > 0$ *inequality holds. Therefore, the linear functions of* α, $x^f(\alpha)$ *and* $s^f(\alpha)$ *do not change sign on the interval* $[0,1]$. *Consequently,* $x^f(0) = x > 0$ *and* $s^f(0) = s > 0$ *yield* $x^f(1) = x^f > 0$ *and* $s^f(1) = s^f > 0$. □

In the following lemmas we analyse how far are the vectors obtained after the feasibility step from the next points of the central path of the perturbed problems.

Lemma 6.2 *Let* $x > 0$ *be a feasible solution of* (P_ν) *and* $s > 0$ *a feasible solution of* (D_ν), *and* $\sigma(v) = \sigma(xs, \mu)$ *such that* $\sigma(v)^2 + 2\omega(v)^2 < 1$. *Then*

$$\sigma(v^+) = \sigma(x^f s^f, \mu_+) \leq \frac{\theta\sqrt{n} + \sigma(v)^2 + 2\omega(v)^2}{1 - \theta + \sqrt{(1-\theta)(1 - \sigma(v)^2 - 2\omega(v)^2)}}.$$

Proof. From Lemma 6.1 we get $x^f > 0$ *and* $s^f > 0$. *Using (18) and the last equation of (13), from (17) we obtain*

$$\frac{1}{\mu}x^f(\alpha)s^f(\alpha) = (1-\alpha)v^2 + \alpha(2v - v^2) + \alpha^2 d_x d_s. \tag{21}$$

Substituting $\alpha = 1$ *into (21) and using (14) we get*

$$\frac{1}{\mu}x^f s^f = (v^f)^2 = 2v - v^2 + d_x d_s = e - (e-v)^2 + d_x d_s = e - \frac{p_v^2}{4} + d_x d_s = e - \frac{q_v^2}{4}.$$

From this we obtain

$$e - (v^f)^2 = \frac{q_v^2}{4}. \tag{22}$$

Using $\sigma(v^+) = \left\| e - \sqrt{\frac{x^f s^f}{\mu_+}} \right\|$ *we get*

$$\sigma(v^+) = \frac{1}{\sqrt{1-\theta}} \left\| \sqrt{1-\theta}e - v^f \right\| = \frac{1}{\sqrt{1-\theta}} \left\| \frac{(1-\theta)e - (v^f)^2}{\sqrt{1-\theta}e + v^f} \right\|.$$

From (22) it follows that

$$\sigma(v^+) = \frac{1}{\sqrt{1-\theta}} \left\| \frac{-\theta e + \frac{q_v^2}{4}}{\sqrt{1-\theta}e + v^f} \right\|$$

and

$$min(v^f) \geq \sqrt{1 - \frac{\|q_v^2\|_\infty}{4}} \geq \sqrt{1 - \frac{\|q_v^2\|}{4}} \geq \sqrt{1 - \frac{\|q_v\|^2}{4}}.$$

Using $\frac{\|p_v\|^2}{4} = \sigma(v)^2$, *we get from (15)*

$$\frac{\|q_v\|^2}{4} = \sigma(v)^2 - d_x^T d_s, \tag{23}$$

and hence

$$min(v^f) \geq \sqrt{1 - \frac{\|q_v\|^2}{4}} = \sqrt{1 - \sigma(v)^2 + d_x^T d_s}. \tag{24}$$

Using (23), (24) and (20) we obtain

$$\begin{aligned}
\sigma(v^+) &= \frac{1}{\sqrt{1-\theta}} \left\| \frac{-\theta e + \frac{q_v^2}{4}}{\sqrt{1-\theta}e + v^f} \right\| \leq \frac{\left\| -\theta e + \frac{q_v^2}{4} \right\|}{\sqrt{1-\theta}(\sqrt{1-\theta} + \sqrt{1 - \sigma(v)^2 + d_x^T d_s})} \\
&\leq \frac{\theta\sqrt{n} + \sigma(v)^2 - d_x^T d_s}{1 - \theta + \sqrt{(1-\theta)(1 - \sigma(v)^2 + d_x^T d_s)}} \\
&\leq \frac{\theta\sqrt{n} + \sigma(v)^2 + 2\omega(v)^2}{1 - \theta + \sqrt{(1-\theta)(1 - \sigma(v)^2 - 2\omega(v)^2)}}.
\end{aligned}$$

This proves the lemma. □

As a consequence we get that Lemma 7 of [2] holds under different assumptions. The next lemma gives an upper bound for $\omega(v)$. In order to accomplish this, we introduce the following notations as in [16]. Let

$$\mathcal{L} = \left\{ \xi \in \mathbb{R}^n : \bar{A}\xi = 0 \right\}$$

be the null space of the matrix $\bar{A}$ and

$$\mathcal{L}^\perp = \left\{ \bar{A}^T y : y \in \mathbb{R}^m \right\}$$

the row space of $\bar{A}$. Note that $\mathcal{L} \perp \mathcal{L}^\perp$, $\mathcal{L} + \mathcal{L}^\perp = \mathbb{R}^n$ and $\mathcal{L} \cap \mathcal{L}^\perp = \{0\}$. The $\left\{ \xi \in \mathbb{R}^n : \bar{A}\xi = \theta\nu r_b^0 \right\}$ affine space is identical with the space $d_x + \mathcal{L}$. We have $d_s \in \theta\nu v s^{-1} r_c^0 + \mathcal{L}^\perp$, hence $d_x + \mathcal{L}$ and $d_s + \mathcal{L}^\perp$ meet in a unique point, which is denoted by q. The following two lemmas can be proved as in [16]. They determine upper bounds for $\omega(v)$ and $\|q\|$.

Lemma 6.3 (cf. Roos [16], Lemma 4.6) *If* $\{q\} = (d_x + \mathcal{L}) \cap (d_s + \mathcal{L}^\perp)$, *then*

$$2\omega(v) \leq \sqrt{\|q\|^2 + (\|q\| + 2\sigma(v))^2}.$$

Lemma 6.4 (cf. Roos [16], Lemma 4.7) *The inequality*

$$\sqrt{\mu}\,\|q\| \leq \theta\nu\zeta \sqrt{e^T \left(\frac{x}{s} + \frac{s}{x} \right)}$$

holds.

In the next lemma we give lower and upper bounds for the components of $\frac{xs}{\mu}$.

Lemma 6.5 *Let* $\sigma(xs, \mu) < \tau < 1$. *Then*

$$\left(1 - \sqrt{\tau}\right)^2 < \frac{x_i s_i}{\mu} < \left(1 + \sqrt{\tau}\right)^2, \quad \textit{for all } 1 \leq i \leq n.$$

Proof. From $\sigma(v) < \tau < 1$ *we obtain* $\|v - e\| < \tau$. *Thus,*

$$(v_i - 1)^2 \leq \sum_{i=1}^{n} (v_i - 1)^2 < \tau, \quad \textit{for all } 1 \leq i \leq n,$$

and this yields

$$1 - \sqrt{\tau} < v_i < 1 + \sqrt{\tau}.$$

This proves the lemma. □

Lemma 6.6 *One has*

$$\sqrt{e^T \left(\frac{x}{s} + \frac{s}{x}\right)} < \frac{1}{\sqrt{\mu}\,(1 - \sqrt{\tau})} \sqrt{\|x\|^2 + \|s\|^2}.$$

Proof. We know that

$$\frac{x_i}{s_i} + \frac{s_i}{x_i} = \frac{x_i^2 + s_i^2}{x_i s_i}, \quad \textit{for all } 1 \leq i \leq n.$$

From Lemma 6.5 we get $x_i s_i > \mu\,(1 - \sqrt{\tau})^2$, *thus*

$$\frac{1}{x_i s_i} < \frac{1}{\mu\,(1 - \sqrt{\tau})^2}.$$

Using this we obtain

$$\frac{x_i^2 + s_i^2}{x_i s_i} < \frac{1}{\mu\,(1 - \sqrt{\tau})^2} \left(x_i^2 + s_i^2\right), \quad \textit{for all } 1 \leq i \leq n,$$

hence

$$e^T \left(\frac{x}{s} + \frac{s}{x}\right) < \frac{1}{\mu\,(1 - \sqrt{\tau})^2} \left(\|x\|^2 + \|s\|^2\right),$$

and this implies the lemma. □

Lemma 6.7 *If* x *and* s *are strictly feasible solutions of* (P_ν) *and* (D_ν) *and* $\sigma(xs, \mu) < \tau$, *then the following inequality holds:*

$$\sqrt{\|x\|^2 + \|s\|^2} \leq \zeta n \left(2 + 2\sqrt{\tau} + \tau\right).$$

Proof. Using that $\bar{x}$ *and* $(\bar{y}, \bar{s})$ *are optimal solutions of the original primal-dual pair, from (6) we get the following system:*

$$\begin{array}{rcll} A\bar{x} & = & b, & 0 \leq \bar{x} \leq \zeta e, \\ A^T \bar{y} + \bar{s} & = & c, & 0 \leq \bar{s} \leq \zeta e, \\ \bar{x}\bar{s} & = & 0. & \end{array} \tag{25}$$

Let y be the vector such that x and (y, s) are the feasible solutions of (P_ν) and (D_ν). Then, using (7) we may write

$$\begin{aligned} Ax &= b - \nu(b - A\zeta e), \qquad x \geq 0, \\ A^T y + s &= c - \nu(c - \zeta e), \qquad s \geq 0. \end{aligned} \tag{26}$$

We follow the method introduced in [16] using the characteristics of the search direction specified by us. However, our approach differs from the one proposed in [16] in the sense that we don't assume that x and (y, s) are perfectly centered. Hence, we have

$$\begin{aligned} A\bar{x} - Ax &= \nu(A\bar{x} - A\zeta e), \qquad x \geq 0, \\ A^T\bar{y} + \bar{s} - A^T y - s &= \nu(A^T\bar{y} + \bar{s} - \zeta e), \qquad s \geq 0. \end{aligned} \tag{27}$$

Therefore

$$\begin{aligned} A\left(\bar{x} - x - \nu\bar{x} + \nu\zeta e\right) &= 0, \qquad x \geq 0, \\ A^T\left(\bar{y} - y - \nu\bar{y}\right) &= s - \bar{s} + \nu\bar{s} - \nu\zeta e, \qquad s \geq 0. \end{aligned} \tag{28}$$

Since the null space and the row space of a matrix are orthogonal we get

$$\left(\bar{x} - x - \nu\bar{x} + \nu\zeta e\right)^T \left(s - \bar{s} + \nu\bar{s} - \nu\zeta e\right) = 0.$$

Let

$$a := (1-\nu)\bar{x} + \nu\zeta e, \quad d := (1-\nu)\bar{s} + \nu\zeta e.$$

Then $(a - x)^T(d - s) = 0$, which implies

$$a^T d + x^T s = a^T s + d^T x.$$

From Lemma 6.5 we get $x_i s_i < \mu\left(1 + \sqrt{\tau}\right)^2$, and this yields

$$x^T s < \mu n \left(1 + \sqrt{\tau}\right)^2.$$

In addition, $\bar{x}^T\bar{s} = 0$, $\bar{x} + \bar{s} \leq \zeta e$ and $\mu = \nu\mu^0 = \nu\zeta^2$. Thus, we may write

$$\begin{aligned} a^T d + x^T s &= \left((1-\nu)\bar{x} + \nu\zeta e\right)^T \left((1-\nu)\bar{s} + \nu\zeta e\right) + x^T s \\ &= \nu(1-\nu)(\bar{x} + \bar{s})^T\zeta e + \nu^2\zeta^2 n + x^T s \\ &\leq \nu(1-\nu)(\zeta e)^T\zeta e + \nu^2\zeta^2 n + \mu n\left(1 + \sqrt{\tau}\right)^2 \\ &= \nu(1-\nu)\zeta^2 n + \nu^2\zeta^2 n + \mu n\left(1 + \sqrt{\tau}\right)^2 \\ &= \nu\zeta^2 n + \mu n\left(1 + \sqrt{\tau}\right)^2 \\ &= \nu\zeta^2 n(2 + 2\sqrt{\tau} + \tau). \end{aligned} \tag{29}$$

Using $a \geq \nu\zeta e$ and $d \geq \nu\zeta e$ it follows that

$$a^T s + d^T x \geq \nu\zeta e^T(x + s) = \nu\zeta(\|x\|_1 + \|s\|_1). \tag{30}$$

Moreover,

$$\|x\|_1 + \|s\|_1 \leq \frac{a^T s + d^T x}{\nu\zeta} = \frac{a^T d + x^T s}{\nu\zeta} \leq \zeta n(2 + 2\sqrt{\tau} + \tau).$$

Since

$$\|x\|^2 + \|s\|^2 \leq (\|x\|_1 + \|s\|_1)^2 \leq \zeta^2 n^2 (2 + 2\sqrt{\tau} + \tau)^2,$$

it follows that $\sqrt{\|x\|^2 + \|s\|^2} \leq \zeta n\,(2 + 2\sqrt{\tau} + \tau)$. *This proves the lemma.* □

In the next lemma we give an upper bound for $\|q\|$, which depends only on θ, n and τ.

Lemma 6.8 *One has*

$$\|q\| < \theta n \frac{2 + 2\sqrt{\tau} + \tau}{1 - \sqrt{\tau}}.$$

Proof. From Lemma 6.7, Lemma 6.6 and $\sqrt{\mu} = \sqrt{\nu}\zeta$ *we get*

$$\sqrt{e^T \left(\frac{x}{s} + \frac{s}{x}\right)} < \frac{\zeta n\,(2 + 2\sqrt{\tau} + \tau)}{\sqrt{\mu}(1 - \sqrt{\tau})} = \frac{n}{\sqrt{\nu}} \frac{(2 + 2\sqrt{\tau} + \tau)}{(1 - \sqrt{\tau})}.$$

From the previous inequality and Lemma 6.4 we obtain

$$\sqrt{\mu}\,\|q\| \leq \theta\nu\zeta \sqrt{e^T \left(\frac{x}{s} + \frac{s}{x}\right)} < \frac{\theta n \sqrt{\nu}\zeta\,(2 + 2\sqrt{\tau} + \tau)}{1 - \sqrt{\tau}}.$$

Since $\sqrt{\mu} = \sqrt{\nu}\zeta$, *the proof of the lemma is complete.* □

Let x^f and s^f be strictly feasible solutions of (P_ν) and (D_ν). Assuming that the value of μ_+ does not change, the vectors x_+ and s_+ can be determined by a full-Newton step at x^f and s^f. Suppose that $\sigma^+ = \sigma(x_+ s_+, \mu_+)$ and we want to show that the algorithm is well defined. We have to specify the values of θ and τ such that after a main iteration the inequality $\sigma^+ < \tau$ holds.

7. Polynomiality of the Algorithm

Now we analyse the consequences of the previous lemmas when $\tau = \frac{1}{16}$ and $\theta = \frac{1}{8n}$.

Corollary 7.1 *If* $\sigma(v) < \frac{1}{16}$ *and* $\theta = \frac{1}{8n}$, *then* $\omega(v) < \frac{1}{2\sqrt{2}}$.

Proof. From Lemma 6.3 and Lemma 6.8 it follows that

$$\begin{aligned} 4\omega(v)^2 &\leq \|q\|^2 + (\|q\| + 2\sigma(v))^2 \\ &< \left(\theta n \frac{2 + 2\sqrt{\tau} + \tau}{1 - \sqrt{\tau}}\right)^2 + \left(\theta n \frac{2 + 2\sqrt{\tau} + \tau}{1 - \sqrt{\tau}} + 2\tau\right)^2. \end{aligned}$$

Using $\tau = \frac{1}{16}$ *and* $\theta n = \frac{1}{8}$ *we get* $4\omega(v)^2 < \frac{1}{2}$, *which implies* $\omega(v) < \frac{1}{2\sqrt{2}}$. □

Corollary 7.2 *If* $\sigma(v) < \frac{1}{16}$ *and* $\theta = \frac{1}{8n}$, *then* x^f *and* s^f *are strictly feasible.*

Proof. Using Lemma 6.1 and Corollary 7.1 we may write

$$\sigma(v)^2 + 2\omega(v)^2 < \frac{1}{16^2} + \frac{1}{4} < 1,$$

so $x^f > 0$ and $s^f > 0$, thus x^f and s^f are strictly feasible. □

Corollary 7.3 *We can define an upper bound for $\sigma(v^+)$. The following inequality holds:*

$$\sigma(v^+) < \frac{1}{4}.$$

Proof. Using Lemma 6.2 we get

$$\begin{aligned}\sigma(v^+) &\leq \frac{\theta\sqrt{n} + \sigma(v)^2 + 2\omega(v)^2}{1-\theta+\sqrt{(1-\theta)(1-\sigma(v)^2-2\omega(v)^2)}} \\ &\leq \frac{\theta\sqrt{n} + \tau^2 + 2\omega(v)^2}{1-\theta+\sqrt{(1-\theta)(1-\tau^2-2\omega(v)^2)}}.\end{aligned}$$

From $\tau = \frac{1}{16}$ and $\theta = \frac{1}{8n}$, using $n \geq 1$, we obtain $\sigma(v^+) < \frac{1}{4}$. □

The following corollary defines an upper bound for the proximity measure.

Corollary 7.4 *Let x_+ and s_+ be the vectors obtained by a full-Newton step at x^f and s^f. Then $\sigma^+ = \sigma(x_+ s_+, \mu_+) < \frac{1}{16}$.*

Proof. We use Lemma 3.2 for (P_ν) and (D_ν), when the initial points are x^f and s^f. □

We call (x, y, s) an ϵ-solution if the following inequality holds

$$max(x^T s, \|b - Ax\|, \|c - A^T y - s\|) < \epsilon.$$

Corollary 7.5 *The algorithm requires at most*

$$8nlog\frac{max\{n\zeta^2, \|r_b^0\|, \|r_c^0\|\}}{\epsilon}$$

iterations.

Proof. Since μ and ν are multiplied by $1-\theta$ at each iteration, we can prove that after at most

$$\frac{1}{\theta}\frac{max\{n\zeta^2, \|r_b^0\|, \|r_c^0\|\}}{\epsilon}$$

inner iterations the algorithm finds an ϵ-solution. The proof is similar to the one of Lemma 3.6. Using $\theta = \frac{1}{8n}$ we obtain the upper bound given in the corollary. □

Since every main iteration contains two inner iterations, we get the following theorem.

Theorem 7.6 *If (P) and (D) are feasible and $\zeta > 0$ is defined such that $\|\bar{x} + \bar{s}\|_\infty \leq \zeta$, where $\bar{x}$ and $(\bar{y}, \bar{s})$ are the optimal solutions of (P) and (D),*

then after at most

$$16nlog\frac{max\{n\zeta^2, \|r_b^0\|, \|r_c^0\|\}}{\epsilon}$$

interior-point iterations the algorithm finds an ϵ-solution of (P) and (D).

8. Numerical experiment

In order to compare the efficiency of infeasible interior-point algorithms with different number of centering steps we have implemented in the C++ programming language a short-step IIPM in the following way. We have used Mehrotra's heuristic [12] followed by a few standard primal-dual steps to define the starting points x^0, y^0 and s^0. In each major iteration the algorithm performs a feasibility step using (10) and one or more centering steps using (4). We calculate the maximum step sizes in order to maintain the nonnegativity of the variables x and s, and we reduce these step sizes with a factor ρ, where $0 < \rho < 1$. As in the case of the algorithm in Figure 2, after the feasibility step we reduce the value of μ and ν by a factor $1 - \theta$, where $0 < \theta < 1$.

The algorithm of Roos [16] performs at most three centering steps in each major iteration. We have solved the problem *afiro*, given in the standard MPS form in the Netlib test collection, using the following parameters: $\epsilon = 0.0001$, $\theta = 0.5$ and $\rho = 0.9999$. The number of major iterations was the same in the case of one centering step and three centering steps. Thus, the algorithm performed 15 major iterations, and a total number of 30 interior-point iterations for one centering step, and 60 iterations for three centering steps. In the algorithm proposed by Roos in each major iteration the centering steps are performed until the proximity measure becomes smaller than the parameter τ. Taking $\tau = 0.25$, the first main iteration of the algorithm performed three centering steps, followed by 4 major iterations with two centering steps and 10 iterations with one centering step. This yields a total number of 36 inner iterations.

9. Conclusion

We have defined a full-Newton step infeasible interior-point algorithm with new search directions. We have applied the square root function for the centering equations and we have used Newton's method in order to get the new search directions. We have proved that the algorithm finds an ϵ-solution in polynomial time. We have shown that in a major iteration it is enough to take only one centering step in order to prove that the algorithm is well defined.

Acknowledgements

We kindly acknowledge the useful comments of the referees. We are also grateful to Ágnes Mester for her suggestions, which improved the presentation of this paper.

References

[1] M. Achache. A new primal-dual path-following method for convex quadratic programming. *Computational and Applied Mathematics*, 25(1): 97–110, 2006.

[2] K. Ahmadi, F. Hasani, and B. Kheirfam. A full-Newton step infeasible interior-point algorithm based on Darvay directions for linear optimization. *Journal of Mathematical Modelling and Algorithms in Operations Research*, 2013. doi: 10.1007/s10852-013-9227-7.

[3] J.F. Bonnans and F.A. Potra. Infeasible path-following algorithms for linear complementarity problems. *Mathematics of Operations Research*, 22(2):378–407, 1997.

[4] Zs. Darvay. A new algorithm for solving self-dual linear optimization problems. *Studia Universitatis Babeş-Bolyai, Series Informatica*, 47(1): 15–26, 2002.

[5] Zs. Darvay. New interior point algorithms in linear programming. *Advanced Modeling and Optimization*, 5(1):51–92, 2003.

[6] G. Gu, H. Mansouri, M. Zangiabadi, Y.Q. Bai, and C. Roos. Improved full-Newton step O(nL) infeasible interior-point method for linear optimization. *Journal of Optimization Theory and Applications*, 145(2): 271–288, 2010.

[7] N.K. Karmarkar. A new polynomial-time algorithm for linear programming. *Combinatorica*, 4:373–395, 1984.

[8] M. Kojima, N. Megiddo, and S. Mizuno. A primal-dual infeasible-interior-point algorithm for linear programming. *Mathematical Programming*, 61 (1-3):263–280, 1993.

[9] I.J. Lustig. Feasibility issues in a primal-dual interior-point method for linear programming. *Mathematical Programming*, 49(1-3):145–162, 1990.

[10] H. Mansouri and C. Roos. Simplified O(nL) infeasible interior-point algorithm for linear optimization using full-Newton steps. *Optimization Methods and Software*, 22(3):519–530, June 2007.

[11] H. Mansouri, T. Siyavash, and M. Zangiabadi. A path-following infeasible interior-point algorithm for semidefinite programming. *Iranian Journal of Operations Research*, 3(1):11–30, 2012.

[12] S. Mehrotra. On the implementation of a primal-dual interior point method. *SIAM Journal on Optimization*, 2(4):575–601, 1992.

[13] S. Pan, X. Li, and S. He. An infeasible primal-dual interior-point algorithm for linear programs based on logarithmic equivalent transformation. *Journal of Mathematical Analysis and Applications*, 314(2):644–660, 2006.

[14] F. A. Potra. A quadratically convergent predictor-corrector method for solving linear programs from infeasible starting points. *Mathematical Programming*, 67(1-3):383–406, 1994.

[15] F. A. Potra. An infeasible-interior-point predictor-corrector algorithm for linear programming. *SIAM Journal on Optimization*, 6(1):19–32, 1996.

[16] C. Roos. A full-Newton step O(n) infeasible interior-point algorithm for linear optimization. *SIAM Journal on Optimization*, 16(4):1110–1136, 2006.

[17] Gy. Sonnevend. An "analytic center" for polyhedrons and new classes of global algorithms for linear (smooth, convex) programming. In A. Prékopa, J. Szelezsán, and B. Strazicky, editors, *System Modelling and Optimization: Proceedings of the 12th IFIP-Conference held in Budapest, Hungay, September 1985*, volume 84 of *Lecture Notes in Control and Information Sciences*, pages 866–876. Springer Verlag, Berlin, West-Germany, 1986.

[18] K. Tanabe. Centered Newton method for linear programming: Interior and 'exterior' point method. In K. Tone, editor, *New Methods for Linear Programming*, volume 3, pages 98–100. 1990. In Japanese.

[19] R.J. Vanderbei. *Linear Programming: Foundations and Extensions*. Kluwer Academic Publishers, Boston, 1996.

[20] G. Q. Wang and Y. Q. Bai. A new full Nesterov-Todd step primal-dual path-following interior-point algorithm for symmetric optimization. *Journal of Optimization Theory and Applications*, 154(3):966–985, 2012.

[21] S.J. Wright. *Primal-Dual Interior-Point Methods*. SIAM, Philadelphia, USA, 1997.

[22] Y. Ye. *Interior Point Algorithms, Theory and Analysis*. John Wiley & Sons, Chichester, UK, 1997.

[23] Y. Zhang. On the convergence of a class of infeasible interior-point methods for the horizontal linear complementarity problem. *SIAM Journal on Optimization*, 4(1):208–227, 1994.

Babeş-Bolyai University, Faculty of Mathematics and Computer Science, No. 1 Mihail Kogălniceanu Street, RO-400084 Cluj-Napoca, Romania
E-mail address: darvay@cs.ubbcluj.ro

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LIX**, Number 1, 2014

AN FCA DRIVEN ANALYSIS OF MAPPING CONCEPTUAL DESIGNS TO XML SCHEMAS

VIORICA VARGA AND CHRISTIAN SĂCĂREA

Abstract. XML is a popular data representation and exchange format over the web. Many articles propose different conceptual models for XML design, but there is no standard format for conceptual design of XML data. Franceschet et al. (2013) demonstrate that nested schemas are globally more efficient than flat ones. In this paper an FCA based representation of XML conceptual modeling is proposed. Entity-Relationship data model is used initially. A Relational FCA approach is given for the relations between entity sets. A mapping from Entity-Relationship model to a schema for XML in form of concept lattices are presented for relations of type one-to-one, one-to-many and many-to-many. The obtained concept lattices reflect these three different relationship types, also the possibility of nested XML scheme design can be seen on the lattices.

1. Introduction and Previous Work

As XML becomes a popular data representation and exchange format over the web, XML schema [7] design has become an important research area. In database environment, the Extended Entity-Relational (EER) model is used as a conceptual schema to represent the structure of the data and the relationships in a relational database, but there is no standard format to represent XML data structure and hierarchies. This article is a contribution to the development of design methodologies for XML databases, advocating the use of Formal Concept Analysis (FCA).

There are many other approaches toward this goal. For instance, Elmasri et al. ([2]) propose a visual XML Schema Designer including EER modeling and UML class diagrams. Fong et al. ([3]) are introducing an XML Tree Model as a conceptual scheme of an XML model. A very recent contribution, by Franceschet et al. [4] propose a mapping from conceptual design to logical

Received by the editors: April 2, 2014.

2010 *Mathematics Subject Classification.* 68P15, 03G10.

1998 *CR Categories and Descriptors.* H.2.1 [**D**atabase Management]: Logical design – Scheme and subschema.

Key words and phrases. XML data design, Formal Concept Analysis.

schemas for XML data. They adopt the Entity-Relationship (ER) model extended with specialization as the conceptual model for native XML databases. Starting with the observation that, in general, the same Entity-Relationship (ER) conceptual schema can be mapped in different XML schemas, they present two alternative ways of mapping ER conceptual schemas into XML: a flat relational-style design methodology and a nesting approach. The former never nests an entity into another entity, but the latter nests entities as much as possible. Also, an experimental evaluation and comparison of these two mapping mechanisms over large datasets is presented. This evaluation shows that both validation and query processing are globally more efficient with nested schemas than with flat ones. This motivates the theoretical problem of finding the best possible nesting for the design.

In this paper a Formal Concept Analysis [5] based representation of XML conceptual modeling is proposed. Entity-Relationship data model is used initially. A Relational FCA approach is given for the relations between entity sets. A mapping from Entity-Relationship model to a schema for XML in form of concept lattices are presented for relations of type one-to-one, one-to-many and many-to-many. The obtained concept lattices [8] reflect these three different relationship types, also the possibility of nested XML scheme design can be seen on the lattices. At this point, Formal Concept Analysis proves to be a valuable tool for the design of such schemas. It gives an expressive graphical representation of the relationships between entities. None of the other approaches for XML scheme design use the two cardinality constraints for binary relationships as Franceschet et al. (2013) [4]. It is a difficult task to understand these two cardinality constraints without a graphical representation. These two cardinality constraints are crucial for nesting. Our approach is based on [4], which is a very recent result, so our solution is basically different from the older proposals. It takes into account two cardinality constraints for binary relationships, it gives an expressive representation of them.

2. Conceptual Hierarchy Representation of Database Conceptual Design

In this section, we propose an FCA grounded approach to conceptual database and XML data design. This problem is definitely not a new one. Even if there are many approaches concept lattices prove to be, once again, the best platform to communicate, understand and discriminate between different conceptual design related notions, due to their expressiveness. This approach is based on the relational context families introduced by Huchard et al. (2007) in [6], as well as on the known notion of a many-valued context.

Definition 1. A relational context family is a pair $(\mathbb{K}, R)$, where

- $\mathbb{K}$ is a set of formal contexts $\mathbb{K}_i = (G_i, M_i, I_i)$, $i \in I$,
- R is a set of binary relations $R_{kl} \subseteq G_k \times G_l$, $k, l \in I$. The contexts $\mathbb{K}(R_{kl}) := (G_k, G_l, R_{kl})$ are called relational contexts.

One of the most common method for conceptual database design is the Entity-Relationship (E-R) model (see [1]), which is a high level conceptual data model and describes in an abstract way the database. An E-R schema consists of entity sets, attributes, and relationships between entity sets. Since attributes are representing properties of real world objects, they are many-valued, hence we can represent every entity set of the E-R Model as a many-valued context.

Let be $E_1, E_2, \ldots, E_l$ be entity sets from the E-R diagram. Every entity set E_i, $i = 1, \ldots, l$ is represented as a many-valued context. If we denote by $A_{i_1}, A_{i_2}, \ldots, A_{i_k}$ the attributes of E_i, the objects are the entities of the entity set E_i and will be denoted by $e_1, e_2, \ldots$. The corresponding values are the values of an entity for a given attribute. We denote the key of the entity set E_i with K_{E_i}.

Entity sets are connected by relations, which represent semantic connections between the entities. The simplest form of relationship is the binary relation. Relationships involving more than two entity sets are called n-ary relations and can equivalently expressed by multiple binary relations. Usually, these relationships are characterized by two roles expressing the function that the two related entities are playing in the relation. These roles can be annotated with maximum cardinality constraints. These constrains are denoting the maximum number of objects, i.e., entities of the codomain of the relation to which any entity of the source set can be related.

Binary relationships have two cardinality constraints ([4]) of the form (x, y), where x is a natural number, that specifies the minimum cardinality or participation constraint. The second cardinality constraint y is the maximum cardinality constraint, which is a positive natural number or N which represents an arbitrarily large natural number.

Let us consider two entity sets A and B and a binary relation R between A and B with left cardinality constraint (x_1, y_1) and right cardinality contraint (x_2, y_2). We use the $A \overset{(x_1,y_1)}{\longleftrightarrow} R \overset{(x_2,y_2)}{\longleftrightarrow} B$ notation for this.

Based on their maximum cardinality constraints, we have

(1) one-to-one relationships, if both roles have maximum cardinality 1;
(2) one-to-many, if one role has maximum cardinality 1 and the other has maximum cardinality N;
(3) many-to-many, if both roles have maximum cardinality N.

We are going to use FCA in order to represent these roles in a concept lattice which will serve as basis of further understanding and communication, as well as for the database conceptual design.

The entity sets A and B can be represented as a many-valued context and every entity from this set being represented as an object in the context. The relationship between entities from A and B will be represented using their entity keys in a formal context.

Definition 2. Let R_{ij} be a relation between the entity sets E_i and E_j, $i, j \in I$. The relational context of R_{ij} is defined as the context having as object sets different key values of K_{E_i} and objects different key values of K_{E_j}, the incidence relation being R_{ij}.

We illustrate the one-to-one relationships in Table 1, the one-to-many in Table 2 and the many-to-many in Table 3. The conceptul lattice representation reflects the conceptual database design, which is independent of database model.

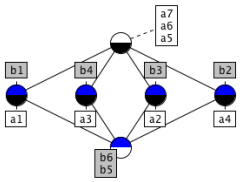
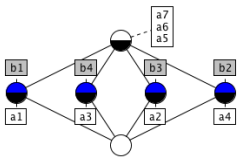
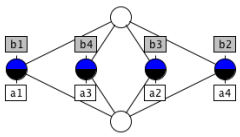
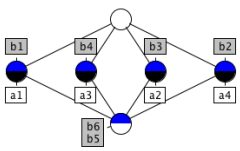
We study also the possibility of mapping the ER Model to the structure of XML data in XSN (introduced in [4]) form. In [4], the authors give two alternative definitions of this mapping, one modeling entities as global XML elements and expressing relationships between them in terms of keys and key references (flat design), the other one modeling relationships by including elements for some entities into the elements for the other entities (nest design). They compare the flat design with the nested one and prove that the nested approach leads to improvements in both query and validation performance.

In the following, we give the nested structure of XML data where it is possible. If the flat structure is the same as the nest structure, no information is provided, since the flat structure has already been studied in the authors previous work.

We denote by KA the key of the entity set A and KB the key of B. In the many-to-many case, nesting of the element for one entity into the element for the other one is never possible. The 0 value for the minimum cardinality constraint specifies that there are elements which are not related to the elements of the other entity set. This determines that the nested design is arguable. We will analyze the different cases below. The representation of an XML binary relationship using concept lattices is more expressive than the classical $A \overset{(x_1,y_1)}{\longleftrightarrow} R \overset{(x_2,y_2)}{\longleftrightarrow} B$ notation. We focus on nested XML data design, where the value of x_1 and x_2 is decisive.

2.1. **One-to-One Relationships.** Let be A and B two entity sets and a R one-to-one relationship between them. For this type of relationship the

TABLE 1. Conceptual Hierarchy Representation of One-to-One Relationship

Nr.	Relationship	Conceptual Hierarchy	Nested XSN
1.	$A \overset{(0,1)}{\longleftrightarrow} R \overset{(0,1)}{\longleftrightarrow} B$		
2.	$A \overset{(0,1)}{\longleftrightarrow} R \overset{(1,1)}{\longleftrightarrow} B$		A (KA, R?) R (B) B(KB)
3.	$A \overset{(1,1)}{\longleftrightarrow} R \overset{(1,1)}{\longleftrightarrow} B$		A (KA, R) R (B) B(KB) OR B (KB, R) R (A) A(KA)
4.	$A \overset{(1,1)}{\longleftrightarrow} R \overset{(0,1)}{\longleftrightarrow} B$		B (KB, R?) R (A) A(KA)

minimum and maximum cardinality constraint values are 0 or 1. If we analyze the lattices of Table 1 we can observe, that for every concept (excepting the top an bottom of the lattice) there exists one element from A and one element from B. In these lattices, the representation of one-to-one relationship is very expressive. The concept lattices vary only in bottom and top elements, but these are decisive in the nested XML scheme. As we can see on Table 1 the nested XML schemes differ in these four cases. In the following, we analyse

the different labeling of the top and bottom elements and hence different types of relations.

Case 1: $A \overset{(0,1)}{\longleftrightarrow} R \overset{(0,1)}{\longleftrightarrow} B$. There are elements of A which are not related to any element of B, and elements of B which are not related to any element of A. They are exactly the intent of the greatest element and the extent of the smallest element of the concept lattice, respectively. Consider for example, the entity set $A = \{a_1, a_2, \ldots, a_7\}$ and $B = \{b_1, b_2, \ldots, b_6\}$as a working example to generate the concept lattice. The elements of A which are not related to elements of B are displayed at the top of the lattice ($\{a_5, a_6, a_7\}$), while the elements of B which are not related to elements of A ($\{b_5, b_6)\}$), appear at the bottom of the lattice. The nested design of XML data is not possible in this case. The inclusion of B in A would lead to the loss of B elements which are not related to elements of A, and vice-versa.

Example: `Men` $\overset{(0,1)}{\longleftrightarrow}$ `spouse` $\overset{(0,1)}{\longleftrightarrow}$ `Women`. One man can be married in christianity with one woman, but there can be unmarried men or women.

Case 2: $A \overset{(0,1)}{\longleftrightarrow} R \overset{(1,1)}{\longleftrightarrow} B$. Consider $A = \{a_1, a_2, \ldots, a_7\}$ and $B = \{b_1, b_2, b_3, b_4\}$. In this case, there are elements of A which are not related to elements of B, and they are displayed at the top of the lattice ($\{a_5, a_6, a_7\}$) as the intent of the greatest concept, but every element of B is related with one element of A. The nested design is possible, including B in A, but not the inverse.

Example 1. `AcademicStaff` $\overset{(0,1)}{\longleftrightarrow}$ `advisor` $\overset{(1,1)}{\longleftrightarrow}$ `StudentsGroups`. Every group of students has one instructor as advisor, but not every staff member is advisor. So StudentsGroup can be nested in AcademicStaff, but not the inverse case. The XSN scheme is the following:

```
AcademicStaff (AName, Gender, BirthDate, advisor?)
  advisor (StudentsGroups)
    StudentsGroups (GroupCode, GroupName?, StudentsNR)
```

Case 3: $A \overset{(1,1)}{\longleftrightarrow} R \overset{(1,1)}{\longleftrightarrow} B$. In this case, every element of A is related to one element of B, and also every element of B is related with one element of A. Consider $A = \{a_1, a_2, a_3, a_4\}$ and $B = \{b_1, b_2, b_3, b_4\}$. The intent of the top element and extent of the bottom element of the concept lattice are empty. In this case both nesting is correct, we can include A in B or B in A.

Example 2. `Faculties` $\overset{(1,1)}{\longleftrightarrow}$ `managed` $\overset{(1,1)}{\longleftrightarrow}$ `Deans`. Every faculty has a dean and every dean manages only one faculty. There are no faculties without dean and no dean who does not manages a faculty. The XSN scheme is the following:

```
Faculties (FName, DeanName, Address, Homepage, managed)
   managed (Deans)
     Deans (AName, Gender, BirthDate)
```

The inverse is correct too.

```
Deans (AName, Gender, BirthDate, managed)
  managed (Faculties)
    Faculties (FName, DeanName, Address, Homepage)
```

Case 4: $A \overset{(1,1)}{\longleftrightarrow} R \overset{(0,1)}{\longleftrightarrow} B$. This case allows the existance of elements of B not related to elements of A, but every element of A is related with one element of B.For $A = \{a_1, a_2, a_3, a_4\}$ and $B = \{b_1, b_2, \ldots, b_6\}$, the are elements of B which are not related to elements of A ($\{b_5, b_6)\}$) are displayed at the bottom of the lattice. This is the inverse of case 2, A can be included in B.

Example 3. `Departments` $\overset{(1,1)}{\longleftrightarrow}$ `managed` $\overset{(0,1)}{\longleftrightarrow}$ `AcademicStaff`. Every department has only one manager, but not every staff member is a manager. The XSN scheme is the following:

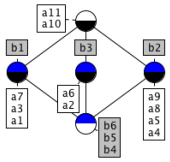
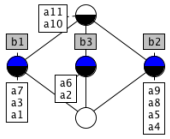
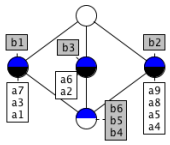
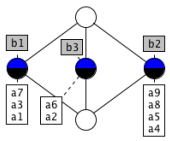
```
AcademicStaff (AName, Gender, BirthDate, managed?)
  managed (Departments)
     Departments (DName, DAddress, DHomepage, worksIn+)
```

2.2. **One-to-Many Relationships.** The R relationship between A and B is a one-to-many relationship. For this type of relationship the minimum cardinality constraint values are 0 or 1 and the maximum cardinality constraint values are 0 or N (an arbitrarily large natural number). If we analyze the lattices of Table 2 we can observe, that for every concept (excepting the top an bottom) exists one element from B and more than one element from A. This illustrates the relationship between elements of A and elements of B. The many part of the one-to-many relationship is A, so on the lattice we can see this assignment between the two entity sets. The XML nested structure includes A (the many part) in B (the one part) if it is possible.

In **case 5**, we have $A \overset{(0,1)}{\longleftrightarrow} R \overset{(0,N)}{\longleftrightarrow} B$. Consider $A = \{a_1, a_2, \ldots, a_{11}\}$ and $B = \{b_1, b_2, \ldots, b_6\}$ as a working example to generate the concept lattice. There are elements of A which are not related to elements of B, being displayed at the top of the lattice ($\{a_{10}, a_{11}\}$), as well as elements of B which are not related to elements of A ($\{b_4, b_5, b_6)\}$), appearing at the bottom of the lattice. Having elements in A not related to a parent, hierarchical design is not possible, thus flat and nest mappings coincide.

Example 4. `Students` $\overset{(0,1)}{\longleftrightarrow}$ `Offered` $\overset{(0,N)}{\longleftrightarrow}$ `Groups`. Let us imagine some groups for students, from which every student can choose 0 or 1 group and

TABLE 2. One-to-Many Relationship Lattice Representation

Nr.	Relationship	Conceptual Hierarchy	Nested XSN
5.	$A \overset{(0,1)}{\longleftrightarrow} R \overset{(0,N)}{\longleftrightarrow} B$		
6.	$A \overset{(0,1)}{\longleftrightarrow} R \overset{(1,N)}{\longleftrightarrow} B$		
7.	$A \overset{(1,1)}{\longleftrightarrow} R \overset{(0,N)}{\longleftrightarrow} B$		B (KB, R*) R (A) A(KA)
8.	$A \overset{(1,1)}{\longleftrightarrow} R \overset{(1,N)}{\longleftrightarrow} B$		B (KB, R+) R (A) A(KA)

in every group can participate 0 or N students. So there can be students which don't choose a group and groups which are empty. The hierarchical design should include students in the selected group, but students can not be included in offered groups, because there can be some students which do not participate in a group.

In **case 6**, we have $A \overset{(0,1)}{\longleftrightarrow} R \overset{(1,N)}{\longleftrightarrow} B$. Take $A = \{a_1, a_2, \ldots, a_{11}\}$ and $B = \{b_1, b_2, b_3\}$. For these cardinality constrains, there are elements of A which are not related to elements of B, as we can see looking at the top of the

lattice ($\{a_{10}, a_{11}\}$), but every element of B is related with elements of A. For the same reason as in case 5, nested design of XML structure is not possible.

Example 5. `Publications` $\overset{(0,1)}{\longleftrightarrow}$ `Has` $\overset{(1,N)}{\longleftrightarrow}$ `PublicationTypes`. We will give some type to publications, like journal paper, conference paper, etc., but there can be some publications which can not be included in one of the PublicationTypes. Every publication type has more publications included, one publication is included in utmost one type. So nested design is not possible. If we try to include Publications in PublicationTypes, then publications which has no type will be lost.

In **case 7**, we have $A \overset{(1,1)}{\longleftrightarrow} R \overset{(0,N)}{\longleftrightarrow} B$. Consider as a working example $A = \{a_1, a_2, \ldots, a_9\}$ and $B = \{b_1, b_2, \ldots, b_6\}$. There are elements of B, which are not related to elements of A ($\{b_4, b_5, b_6)\}$), appearing in the bottom of the lattice, but every element of A is related with one element of B. Hierarchical design of XML data is possible, including A in B. There are elements of B without child elements and this is expressed by the notation $R*$.

Example 6. `Specializations` $\overset{(1,1)}{\longleftrightarrow}$ `SpecOfferedBy` $\overset{(0,N)}{\longleftrightarrow}$ `Faculties`. Every specialization is included only in one faculty, one faculty has more specializations included, but there can be faculties which are not divided in specializations. So nested design is possible, specializations are included in corresponding faculties. The XSN scheme is the following:

```
Faculties (FName, DeanName, Address, Homepage, SpecOfferedBy*)
   SpecOfferedBy (Specialization)
      Specialization (SpecName, AcceptedStudentsNR)
```

`Faculties` without specializations will have no `SpecOfferedBy` elements.

In **case 8**, we have $A \overset{(1,1)}{\longleftrightarrow} R \overset{(1,N)}{\longleftrightarrow} B$. For these constrains, every element of A is related to one element of B, and also every element of B is related with elements of A. The intent of the top element and the intent of the bottom element are empty. The concept lattice has been generated for $A = \{a_1, a_2, \ldots, a_9\}$ and $B = \{b_1, b_2, b_3\}$. Nested structure of XML data is possible, there are no parent elements without child elements and we denote this by $R+$.

Example 7. `AcademicStaff` $\overset{(1,1)}{\longleftrightarrow}$ `worksIn` $\overset{(1,N)}{\longleftrightarrow}$ `Departments`. Every staff member works in a department, but in one department there are more staff members. There are no departments without members and there are no members which are not in some department. The XSN scheme is the following:

```
Departments (DName, DAddress, DHomepage, worksIn+)
  worksIn (AcademicStaff)
    AcademicStaff (AName, Gender, BirthDate)
```

2.3. **Many-to-Many Relationships.** In this paragraph we study the many-to-many relationships. In contrast with concept lattices for one-to-one and one-to-many relationships, if we analyze the lattices displayed in Table 3 we can observe that the structure of the concept lattice is more complex. For one-to-one and one-to-many relationships they are just nominal scales with additional information at the top and bottom element of the lattice. If the relationship is of type one-to-one concepts in every level are labeled only with attribute (the key of one element from entity set B) and one object (the key of one element from entity set A). If the relationship is of type one-to-many, then the the concepts are labeled by one attribute (the key of one element from entity set B, the parent) and more objects (the keys of elements from entity set A, the child). Concepts of conceptual lattices which represent many-to-many relationships are labeled by more attributes (the keys of elements from entity set B) and more objects (the keys of elements from entity set A) and these concepts are on more hierarchical levels. The top and bottom of the concept lattice are labeled if there are dangling (they are not related by relation R to the other entity set) elements in the entity sets A and B. In cases 9, 10, and 11 there are dangle elements, which are displayed at the bottom and the top of the lattice. Nested design is not possible for many-to-many relationships, so we will not analyze in detail this kind of relationship.

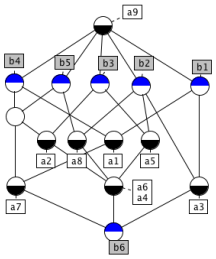
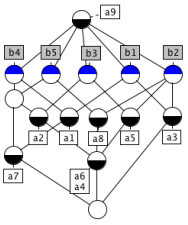
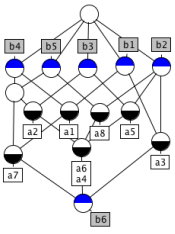
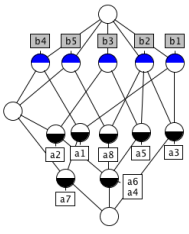
Example 8. `Students` $\overset{(0,N)}{\longleftrightarrow}$ `choose` $\overset{(0,N)}{\longleftrightarrow}$ `Courses`. One course can be chosen by 0 or more students and there can be students which choose 0 or more courses. This is an example for case 9. This example can be considered for case 10 as well, but with different constraints: every course was chosen by 1 or more students. The example can be similarly adapted for case 11 and 12.

3. Conclusion and Further Work

This article is a contribution to XML data design. It is a new proposal since it is based on Franceschet et al. (2013) [4]. Concept lattices give an expressive graphical representation of the relationships between entities. Our approach uses two cardinality constraints for binary relationships as Franceschet et al. [4]. The representation of XML data relationships using concept lattices proves to be not only expressive, but also very useful, especially for the 0 minimum cardinality constraints.

As future work we propose to implement a graphical tool, which give the possibility to construct Entity-Relationship Model with minimum and maximum constraints on binary relationship. The software will build the conceptual lattice from E-R model and the nested or flat XML scheme will be proposed.

TABLE 3. Many-to-Many Relationship Lattice Representation

Nr.	Relationship	Conceptual Hierarchy
9.	$A \overset{(0,N)}{\longleftrightarrow} R \overset{(0,N)}{\longleftrightarrow} B$	
10.	$A \overset{(0,N)}{\longleftrightarrow} R \overset{(1,N)}{\longleftrightarrow} B$	
11.	$A \overset{(1,N)}{\longleftrightarrow} R \overset{(0,N)}{\longleftrightarrow} B$	
12.	$A \overset{(1,N)}{\longleftrightarrow} R \overset{(1,N)}{\longleftrightarrow} B$	

References

[1] R. Elmasri, S.B. Navathe: *Fundamentals of Database Systems*, Addison Wesley, (2010).
[2] R. Elmasri et al.: *Conceptual modeling for customized XML schemas*, Data and Knowledge Engineering 54, pp. 57-76 (2005).
[3] J. Fong, S. K. Cheung, H. Shiu: *The XML Tree Model toward an XML conceptual schema reversed from XML Schema Definition*, Data and Knowledge Engineering 64, pp. 624-661, (2008)
[4] M. Franceschet, D. Gubiani, A. Montanari, C. Piazza: *A Graph-Theoretic Approach to Map Conceptual Designs to XML Schemas*, ACM Transactions on Database Systems, Vol. 38, pp. 6-44 (2013)
[5] B. Ganter, R. Wille.: *Formal Concept Analysis*. Mathematical Foundations. Springer, Berlin-Heidelberg-New York(1999)
[6] M. Huchard, M.R. Hacene, C. Roume, P. Valtchev.: *Relational concept discovery in structured datasets*, Ann. Math. Artif. Intell. 49(1-4) (2007) pp. 39-76
[7] W3C. *XML Schema*, http://www.w3.org/TR/xmlschema-0/ (2004)
[8] S.A. Yevtushenko: *System of data analysis "Concept Explorer"*. (In Russian). Proceedings of the 7th national conference on Artificial Intelligence KII, Russia, pp. 127-134 (2000).

Department of Computer Science, Faculty of Mathematics and Computer Science, Babeş-Bolyai University, Kogălniceanu 1, 400084 Cluj-Napoca, Romania
E-mail address: ivarga@cs.ubbcluj.ro
E-mail address: csacarea@math.ubbcluj.ro

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LIX**, Number 1, 2014

ON EVALUATING THE STRUCTURE OF SOFTWARE PACKAGES

ZSUZSANNA MARIAN

Abstract. In this paper we present a study on how a measure previously introduced in the literature can be used to evaluate the structure of software packages in a software system. Three open-source case studies are used and for each case study the value of this measure for four different divisions into packages is investigated. For these case studies we compute the value of other metrics from the literature as well, and a comparison on how they evaluate these divisions is given. We conclude about the relevance of the analyzed measure for evaluating the quality of a partitioning into packages.

1. Introduction

Modern software systems are increasingly complex, made of a great number of different components, which are usually organized into some kind of hierarchical structures. For example, in case of object-oriented systems, the components of the system are the classes, which are organized into packages. Finding how exactly should classes be divided between the packages is not at all a trivial problem, since many different criteria have to be considered, when designing these packages. Usually, the main criterion is related to the dependencies between the classes, but others can be used as well: how similar the names of the classes are, what other classes are used by the classes in the package, and so on.

How classes are organized into packages depends also on the architecture of the software system. There is a general rule, "low coupling, high cohesion", but this cannot be applied for every software system. While in case of frameworks this rule is true, one wants separate, loosely coupled packages, in case of a layered architecture there are relations between elements belonging to different

Received by the editors: April 2, 2014.

2010 *Mathematics Subject Classification.* 68N30.

1998 *CR Categories and Descriptors.* D.2.8 [**Software Engineering**]: Metrics – Performance measures; D.2.10 [**Software Engineering**]: Design.

Key words and phrases. software engineering, software package, evaluation measure.

layers, and classes belonging to visualization and business layers should not be placed in the same package, just to reduce coupling between packages.

In [11] we have proposed a novel clustering-based [9] method, called *HASP*, for restructuring classes into packages in a software system. Our method, applicable in case of frameworks, is based on the value of several features, aggregated into a single score, which was used as a distance measure during the clustering process. In the same paper we have introduced a second measure, which can evaluate how well-structured a software system is, with respect to an apriori known good structure. In this paper we present a study performed in order to evaluate how well packages in a software system are structured using the evaluation measures introduced in [11]. In our study we will consider and evaluate four different package structures for three open-source software systems using the above-mentioned measures.

The rest of this paper is structured in the following way: Section 2 presents a background on package structure measures existing in the literature. Section 3 represents the experimental part of our paper, consisting of the description of the case studies and of the performed experiments. The results of the experiments, their analysis and a comparison to other metrics presented in the literature are given in Section 4. Finally, Section 5 concludes the paper and outlines some further research directions.

2. Background

In this section we will briefly present the evaluation measure previously introduced in [11] that will be used in our experiments to evaluate how well a software system is divided into software packages. A brief review of other metrics presented in the literature to measure how well the packages of a software system are structured will also be presented.

2.1. **Evaluation measure for software package structures.** In [11] we have introduced a new evaluation measure (Formula 1), which measures how well one package is structured in a software system. In order to formally define this measure, we will consider that a software system S is a set of application classes, $S = \{s_1, s_2, ..., s_n\}$. The set $\mathcal{K} = \{K_1, K_2, ..., K_v\}$ is called a partition into packages of the software system S iff:

- $1 \leq v \leq n$;
- $K_i \subseteq S,\ K_i \neq \emptyset,\ \forall i,\ 1 \leq i \leq v$;
- $\bigcup_{i=1}^{v} K_i = S,\ K_i \bigcap K_j = \emptyset,\ \forall i, j,\ 1 \leq i, j, \leq v,\ i \neq j$.

In order to characterize how well-structured a package K_i is, with respect to a partition $\mathcal{K}$, we have defined five features. In defining these features we

have concentrated on characterizing packages from frameworks, and tried to go beyond the simple "low coupling, high cohesion" principle. The five features for a package P, as presented in [11], are the following:

- F_1 **- Package cohesion** - measures how cohesive the classes inside a package P are, by counting the dependencies between elements of the package P.
- F_2 **- Package reuse** - measures how many packages from the system depend on package P.
- F_3 **- Package coupling** - measures on how many packages from the system does the package P depend on.
- F_4 **- Name cohesion** - measures how similar the names of the classes in package P are.
- F_5 **- Dependency similarity** - measures how similar the classes on which classes from package P depend on are.

Using the above presented five features we have defined a measure to evaluate the division into packages of a whole software system. For a partition $\mathcal{K}$ of a software system, this measure, called *overallScore*, is computed in the following way [11]:

$$overallScore(\mathcal{K}) = \frac{\sum_{i=1}^{v} sc(K_i, \mathcal{K})}{v} \tag{1}$$

where sc is a score that denotes how well-structured a package K_i is, with respect to the partition $\mathcal{K}$, and is computed as [11]:

$$sc(K_i, \mathcal{K}) = \begin{cases} \frac{\sum_{i=1}^{2} w_i * F_i - w_3 * F_3}{|K_i|^2 - 1} + \sum_{i=4}^{5} w_i * F_i, & if\ |K_i| > 1 \\ 0, & otherwise \end{cases} \tag{2}$$

In Formula (2) F_i $(1 \leq i \leq 5)$ denotes the features presented above, while w_i $(1 \leq i \leq 5)$ represents some weights associated to these features.

2.2. **Literature Review.** There are several methods presented in the literature for evaluating the quality of a software package. A set of 13 such metrics is presented by Sarkar et al. in [13]. They consider a large set of possible relations between classes and methods, and define metrics which measure, for example, module interaction, intermodule coupling, association-induced coupling, and so on. Unfortunately, their metrics have a disadvantage: they consider that each module from a software system has some declared APIs (which can be of

two types, Service API and Extension API), but there are many cases, when such APIs are not defined, so their metrics cannot be computed.

This disadvantage is overcome in [6] and [10], where a set of non-API based metrics are proposed for evaluating the remodularization of a software system. Ducasse et al. firstly define the modularity principles on which their metrics are based (for example, information hiding, encapsulation and changeability) then introduce seven metrics that measure these principles. The metrics are based on two different types of dependencies between the packages (constructed from dependencies between the classes from the packages): extension and usage.

A different set of package metrics is introduced by Abreu et al. in [7], not just for characterizing the modularization of a software systems, but also for remodularizing it in a better structure, using a clustering-based approach. They consider 12 different kinds of relations/coupling types between classes, such as: direct inheritance, attribute type, return in operation, local attribute, and so on, but also claim that only minimizing coupling and increasing cohesion is not sufficient for achieving a good package structure.

Ponisio et al. in [12] propose one single cohesion metric for packages, called *Common-Use*. In order to compute it, they consider four types of dependencies/interactions: inheritance, state, class reference and message sends. The main idea behind the *Common-Use* metric is that if all clients of a package use all classes from the package together, then the package is cohesive, even if the classes from the package do not use each other directly.

3. Main results

In this section we will describe the experiments that we have performed in order to evaluate different package structures for software systems. More precisely, for three different software systems (described in Section 3.2) we will evaluate four different package structures using the *overallScore* measure presented in Section 2.1.

3.1. **The CIP measure.** Besides *overallScore*, we have introduced in [11] another measure, called *CIP - Cohesion of Identified Packages*, which adapts the measure introduced in [8] for evaluating the results of aspect mining techniques. The value of *CIP* measures how close one partition (one division of classes into packages) is to another partition. More exactly, given a partition known to be correct, denoted by $\mathcal{K}^{good}$, and another partition, denoted by $\mathcal{K}$, $CIP(\mathcal{K}^{good}, \mathcal{K})$ measures the cohesion of the packages from $\mathcal{K}^{good}$ in $\mathcal{K}$. *CIP*

is computed in the following way [11]:

$$CIP(\mathcal{K}^{good}, \mathcal{K}) = \frac{1}{q} \sum_{i=1}^{q} cip(K_i^{good}, \mathcal{K}). \tag{3}$$

where q is the number of packages in the partition $\mathcal{K}^{good}$, K_i^{good} is the i-th package in $\mathcal{K}^{good}$ and $cip(K_i^{good}, \mathcal{K})$ measures the cohesion of package K_i^{good} in partition $\mathcal{K}$ defined as:

$$cip(K_i^{good}, \mathcal{K}) = \frac{\sum\limits_{k \in M_{K_i^{good}}} \frac{|K_i^{good} \cap k|}{|K_i^{good} \cup k|}}{|M_{K_i^{good}}|} \tag{4}$$

where $M_{K_i^{good}}$ is:

$$M_{K_i^{good}} = \{k | k \in \mathcal{K}, K_i^{good} \cap k \neq \emptyset\} \tag{5}$$

The value of the *CIP* measure is always between 0 and 1, the value of 1 being achieved when $\mathcal{K}$ coincides with the good structure $\mathcal{K}^{good}$. The higher the value of *CIP*, the better the structure of $\mathcal{K}$ with respect to $\mathcal{K}^{good}$.

3.2. **Case studies.** Since both the *overallScore* measure and the five features presented in Section 2.1 were defined in order to evaluate software systems which are frameworks, we have chosen three different open-source frameworks for this case study. All three systems are part of the *Apache Commons* [4] project.

The first case study is the *DbUtils* framework, version 1.5, available at [1], which is a small set of classes designed to make working with JDBC easier. It consists of 25 classes, divided into three packages: *handlers*, *wrappers* and the *default* package.

The second case study is the *Email* framework, version 1.3.2, available at [3], which is a framework for sending emails, built on top of the Java Mail API, but tries to simplify it. It consists of 19 classes, divided into three packages: *resolver*, *util* and the *default* package.

The third case study is the *EL* framework, version 1.0, available at [2], a JSP 2.0 Expression Language interpreter. It consists of 57 classes, divided into two packages: *parser* and the *default* package.

The reasons for choosing these software systems as case studies are the following:

- All three of them are frameworks, which is very important, since the *overallScore* measure was defined for frameworks.
- They are openly available.

- They have a relatively small number of classes, which allows manual verification and analysis.

3.3. **Experiments.** Through the performed experiments we aim at emphasizing that the *overallScore* measure is well-correlated with the *CIP* measure. Thus, instead of *CIP*, which requires the apriori knowledge of a good partition, *overallScore* can be used for evaluating a software package structure.

For all three open-source cases studies presented in Section 3.2 we have considered four different package structures, created in the following way:

- **Original** - is the original package structure for the system.
- **HASP** - the package structure indicated by our *HASP* approach introduced in [11].
- **WP1** - a "wrong partition" created by taking some classes from the packages in the original package structure and moving them into one or more newly created packages. In this way, part of the original structure is kept, but new, incorrect, packages are created.
- **WP2** - a second "wrong partition" created by dividing randomly all the classes into packages.

During the experimental evaluation we have computed the value of the *overallScore* measure for each partition. We have also computed the value of the *CIP* measure for each partition. When computing the *CIP* measure, one has to provide a correct partition, $\mathcal{K}^{good}$. Out of the four partitions, *WP1* and *WP2* cannot be considered as a correct partition, so we had to decide between the original partition and the one provided by our algorithm. After an analysis of both partitions, we have decided to consider the partition provided by the *HASP* approach as the correct one for the *DbUtils* and the *EL* system, while for the *Email* system we have computed the *CIP* measure twice, once considering as $\mathcal{K}^{good}$ the original partition and once the *HASP* partition. In the following these values will be denoted by CIP_O and CIP_H, respectively. Our reasons for these decisions are:

- For the *DbUtils* system we have explained in detail in [11] why we consider the *HASP* partition better than the original one.
- In case of the *Email* system, there are four classes which are placed in a different package in the *HASP* partition than in the original one. Out of these four classes we consider that two were moved correctly (they should rather be in that package), but the other two were better placed in their original package. This is why we consider both the original and the *HASP* partition as a possibly correct partition when computing the value of the *CIP* measure.

- In case of the *EL* system, there is quite a big difference between the original and the *HASP* partition. The original partition has two packages, while the *HASP* partition has seven. Out of these seven packages, one corresponds exactly with the *parser* package from the original partition, while the remaining six packages contain the classes which were originally in the *default* package. Out of these classes, divided in the six packages, our analysis concluded that six classes are not placed correctly. Even with these misplaced classes, we consider that this structure is better than one huge package containing 50 classes.

For computing the *overallScore* measure, values for the weights from Formula (2) are needed. In [11] we have described a grid-search process, through which we identified good values for the weights. Thus, for the experiments presented in this paper, we used the weights reported in [11], namely: $w_1 = 0.22$, $w_2 = 0.25$, $w_3 = 0.2$, $w_4 = 0.52$ and $w_5 = 0.72$.

4. Results and discussion

For all three open-source case studies presented in Section 3.2 we have computed both the *overallScore* and the *CIP* measure, for all four package structures described in Section 3.3. The results are presented in Tables 1, 2 and 3.

Analyzing the values in these tables we can observe that in case of the *overallScore* measure the two "wrong partitions", *WP1* and *WP2*, have lower *overallScores* than the two other partitions. Moreover, *WP2*, the partition where the classes were randomly assigned to the packages, has always the lowest *overallScore* value. Considering the *Original* and the *HASP* partitions, we can see that in case of the first two projects, the *Original* partition has higher *overallScore*, while for the *EL* project, *overallScore* is higher for the *HASP* partition.

Analyzing the values of the *CIP* measure we can see that it always has the value 1 for the *HASP* partition, which is caused by the fact that we consider the *HASP* partition the correct one, with the exception of the *Email* system where

Partition	**OverallScore**	**CIP**
Original	0.9009	0.4183
HASP	0.7349	1
WP1	0.5658	0.4146
WP2	0.2436	0.1372

Table 1. Values for the *overallScore* and *CIP* measures for the *DbUtils* system.

Partition	**OverallScore**	CIP_O	CIP_H
Original	1.146	1	0.6874
HASP	0.9696	0.6688	1
WP1	0.8335	0.54	0.4895
WP2	0.4026	0.1781	0.1912

TABLE 2. Values for the *overallScore* and *CIP* measures for the *Email* system.

Partition	**OverallScore**	**CIP**
Original	0.5241	0.2857
HASP	0.8261	1
WP1	0.3752	0.2672
WP2	0.1601	0.1299

TABLE 3. Values for the *overallScore* and *CIP* measures for the *EL* system.

we consider as the correct partition the *Original* one as well. For *DbUtils*, *EL* and CIP_H for *Email* (i.e., the partition provided by the *HASP* algorithm for the *Email* system), the order of partitions given by the *CIP* measure is the same: the *Original* partition has the second highest value, *WP1* the third one and finally, *WP2* has the lowest value. In case of CIP_O for *Email* (i.e., the original partition for the *Email* system), the *Original* partition has the highest value and the *HASP* partition has the second highest value.

In order to see how correlated these values are, we have computed two rank-based correlation measures between the *overallScore* and the *CIP* values: the Spearman correlation [14], [15] and Spearman's footrule [5], which is computed as the sum of the absolute values of the differences between the ranks of the elements. These values are presented in Table 4. Since for the *Email* system we computed two *CIP* values, we will have two lines with correlation values as well. From this table we can see that the values for the correlations are quite high, especially for the *EL* system and CIP_O for *Email*, both having perfect values. The lowest correlation values are achieved for the *DbUtils* system and CIP_H for *Email*, but even these values show a strong positive correlation between the two measures. In case of Spearman's footrule a lower value corresponds to a better association, so the values for Spearman's footrule from Table 4 show a strong positive correlation of *overallScore* and *CIP* as well.

Project	Spearman correlation	Spearman's footrule
DbUtils	0.8	2
Email - CIP_O	1	0
Email - CIP_H	0.8	2
EL	1	0

TABLE 4. Correlations between *overallScore* and *CIP* for the three case studies.

In order to compare our evaluation measure with other measures from the literature, we computed the value of the seven metrics introduced in [6]. These metrics are:

- IIPU - Index of Inter-Package Usage.
- IIPE - Index of Inter-Package Extending.
- IPCI - Index of Package Changing Impact.
- IIPUD - Index of Inter-Package Usage Diversion.
- IIPED - Index of Inter-Package Extending Diversion.
- PF - Package Focus.
- IPSC -Index of Package Services Cohesion.

For each metric, we considered as extending relation the implementation of interfaces as well. Each metric takes values in the $[0, 1]$ interval, according to [6], higher values correspond to better package structures. The values of these metrics for the four partitions of the three case studies are presented on Tables 5, 6 and 7.

Analyzing the values from Tables 5, 6 and 7 we can observe that generally the same tendency is followed as for the *overallScore* and *CIP* measures: out of the total of 21 cases, in 20 the smallest value was for the *WP2* partition (but in the remaining one case it has a value of 1, which means a perfect structure). Moreover, in 16 cases, the values for the *Original* and the *HASP* partitions are higher than the ones for *WP1* and *WP2*.

Partition	IIPU	IIPE	IPCI	IIPUD	IIPED	PF	IPSC
Original	0.4286	0.5333	0.8333	1	1	1	1
HASP	0.2857	1	0.8333	1	1	1	1
WP1	0.1429	0.6667	0.65	0.9	0.9	0.8333	0.9667
WP2	0.1429	0.1333	0.15	0.8	0.5333	0.7667	1

TABLE 5. Values for metrics from [6] for the *DbUtils* system.

Partition	IIPU	IIPE	IPCI	IIPUD	IIPED	PF	IPSC
Original	0.8889	1	0.8333	1	1	1	1
HASP	0.7778	1	0.8333	1	1	1	1
WP1	0.6667	0.5	0.7	1	0.9	0.9	1
WP2	0.2222	0.25	0.25	0.9167	0.75	0.6806	0.9167

TABLE 6. Values for metrics from [6] for the *Email* system.

Partition	IIPU	IIPE	IPCI	IIPUD	IIPED	PF	IPSC
Original	0.9362	1	0.5	1	1	1	1
HASP	0.3404	0.8571	0.7381	0.8881	0.9286	0.9524	0.9841
WP1	0.4894	0.3143	0.4167	0.9167	0.625	0.8125	0.9688
WP2	0.1064	0.1714	0.3	0.6347	0.5	0.4972	0.8901

TABLE 7. Values for metrics from [6] for the *EL* system.

Still, there are a lot of cases, when the values of a metric are the same for more partitions, in nine cases two partitions have the same value (usually *Original* and *HASP*, having a value of 1) and in three cases there are three equal values, which shows that the metrics can not always differentiate between two partitions. Moreover, we have computed the values of these seven metrics for a fifth partition, one where all classes are in the same package. We did not use this partition for the rest of the experiments, because it is a trivial one, but in case of these seven metrics we noticed that when all classes are in the same package, the value of these metrics is 1, suggesting a perfect "modularization".

To compare the value of these metrics with the value of the *overallScore* measure, we have decided to compute the value of Spearman's footrule for the *overallScore* measure and some of the metrics. In order to avoid having equal ranks we have only considered those metrics which had four distinct values for the partitions. Thus, for the *DbUtils* system we have only considered the IIPE metric, for *Email* only the IIPU metric and for the *EL* system we considered all seven of them. The values for Spearman's footrule for these projects are presented on Table 8.

A visual comparison of the footrule is presented on Figure 1, where dark gray bars represent the footrule values between the *overallScore* and *CIP* measures (presented in Table 4), while the light gray bars represent footrule values between *overallScore* and metrics from [6] (presented in Table 8). In case of systems where we had multiple footrule values (there were two values for the *Email* system in Table 4 and there were seven different values for *EL* in Table 8) we used the average of the values. From Figure 1 we can see that for the

Project	Metric	Spearman's footrule
DbUtils	IIPE	4
Email	IIPU	0
EL	IIPU	4
	IIPE	4
	IPCI	0
	IIPUD	4
	IIPED	2
	PF	2
	IPSC	2

TABLE 8. Spearman's footrule for the case studies between the value of *overallScore* and some metrics from [6]

DbUtils and *EL* systems the light gray bars are a lot higher than the dark gray ones, meaning that the footrule values between *overallScore* and *CIP* are a lot better. In case of the *Email* system this is reversed, the footrule value between *overallScore* and other metrics is lower.

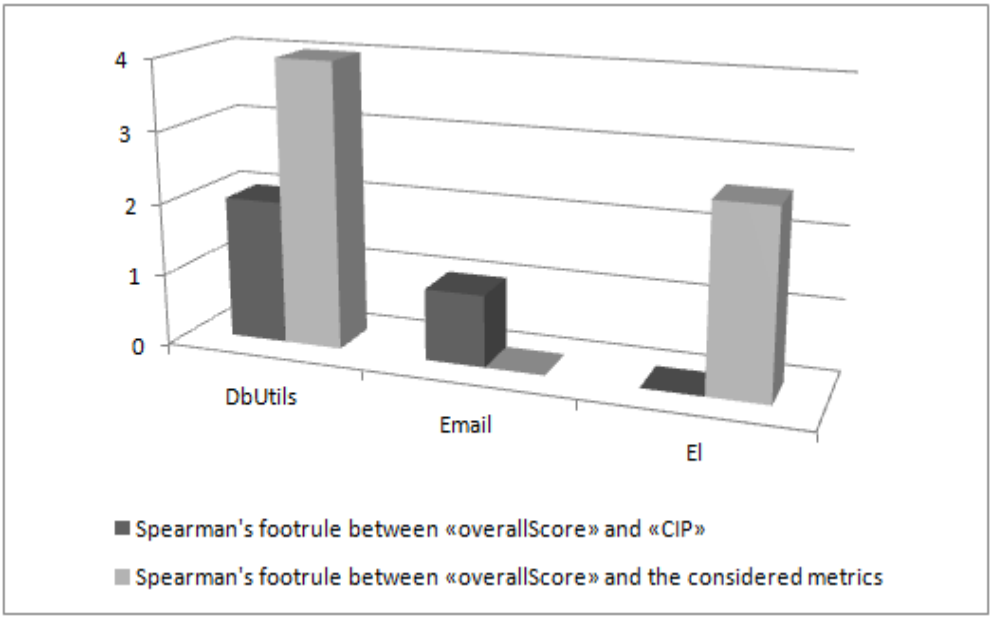


FIGURE 1. Comparison of footrule values.

Considering the comparison of the footrule values presented in Figure 1 and the fact that the considered metrics, presented in [6], often return equal values for different partitions, we can conclude that the *overallScore* measure is more suitable for evaluating a partition of a software system than the other metrics. Moreover, since the values of the *overallScore* and *CIP* measures are strongly positively correlated, as presented in Table 4, *overallScore* can be used instead of *CIP*. This is important, since *CIP* measures how close a partition is to an apriori known correct partition, but when one has to restructure into packages a software system, the good partition is not known. Since the values of the two measures are correlated, *overallScore* can be used, instead of *CIP*, to evaluate a partition.

5. Conclusions and further work

In this paper we have presented a study on the value of two measures designed to evaluate different modularizations of a framework software system. We have considered three open-source case studies and four different partitions for each of them and computed the value of two measures, *overallScore* and *CIP* for them. These two measures were used to automatically restructure a framework into packages. We have also computed the values of other metrics presented in the literature that measure modularization of a software system for these case studies.

Analyzing the values of these metrics and the correlations between them, we have observed that our measures, *overallScore* and *CIP*, are capable of differentiating between a good and a bad partitioning of a software system, which did not always happen in case of the other metrics taken from the literature.

As further work, we would like to perform experiments on other open-source case studies, and compare our measures to other metrics reported in the literature, besides the ones used in this paper. Since our *overallScore* measure was defined for software systems which are frameworks, we would also like to define such a score for systems with other architectures.

References

[1] Dbutils. http://commons.apache.org/proper/commons-dbutils/.
[2] El. http://commons.apache.org/proper/commons-el/.
[3] Email. http://commons.apache.org/proper/commons-email/.
[4] Apache commons project. http://commons.apache.org/.
[5] Persi Diaconis and R. L. Graham. Spearman's footrule as a measure of disarray. *Journal of the Royal Statistical Society*, 39(2):262–268, 1977.
[6] Stéphane Ducasse, Nicolas Anquetil, Usman Bhatti, and Andre Cavalcante Hora. Software metrics for package remodularization. Technical report, Institut National de Recherche en Informatique et en Automatique, 2011.

[7] Fernando Brito e Abreu and Miguel Goulão. Coupling and cohesion metrics as modularization drivers: Are we being over-persuaded? In *Software Maintenance and Reengineering, 2001. Fifth European Conference on*, pages 47–57, 2011.

[8] Istvan Gergely Czibula Gabriela Czibula, Grigoreta Sofia Cojocar. Evaluation measures for partitioning based aspect mining techniques. *International Journal of Computers, Communications & Control*, VI(1):72–80, 2011.

[9] Jiawei Han. *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2005.

[10] Houari Sahraoui Hani Abdeen, Stéphane Ducasse. Modularization metrics: Assessing package organization in legacy large object-oriented software. In *International WorkingConference on Reverse Engineering*, pages 394–398, 2011.

[11] Zsuzsanna Marian, Gabriela Czibula, and Istvan Gergely Czibula. Software packages refactoring using a hierarchical clustering-based approach. *Information Systems*, 2014. Under review.

[12] Laura Ponisio and Oscar Nierstrasz. Using contextual information to asses package cohesion. Technical report, 2006.

[13] Santonu Sarkar, Avinash C. Kak, and Girish Maskeri Rama. Metrics for measuring the quality of modularization of large-scale object-oriented software. *IEEE Transactions on Software Engineering*, 34(5):700–720, 2008.

[14] C. Spearman. The proof and measurement of association between two things. *Amer. J. Psychol.* ***15***, pages 72–101, 1904.

[15] Kelly H. Zou, Kemal Tuncali, and Stuart G. Silverman. Correlation and simple linear regression. *Radiology*, (227):617–622, 2003.

Department of Computer Science, Faculty of Mathematics and Computer Science, Babeş-Bolyai University, Kogălniceanu 1, 400084 Cluj-Napoca, Romania
E-mail address: marianzsu@cs.ubbcluj.ro

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LIX**, Number 1, 2014

DATA TRANSFER OPTIMIZATION IN DISTRIBUTED DATABASE QUERY PROCESSING

LEON ŢÂMBULEA, ADRIAN SERGIU DĂRĂBANT, AND VIORICA VARGA

Abstract. Query execution in a distributed database requires data transfers between the processing nodes of the system. An important step in the query optimization is the minimization of the data transfers which often incur larger latencies than local data processing. In this paper we propose a new method and algorithms for determining the processing nodes for the evaluation of each relational operator of a query so that data transfer is minimal. We model our method as a data transfer minimal cost problem.

1. Introduction

Distributed database system design and query processing is an active research area. The main topics are fragmentation, the allocation of the fragments to various sites, generation of subqueries for sites. Query processing includes designing algorithms that analyze queries and convert the queries into a set of data manipulation operations. An important aspect of query processing is query optimization. The distributed query optimization problem is NP-hard ([10]), which makes finding efficient solution methods and effective heuristics a high priority. Good surveys on query optimization can be found in [5], [11], [20] and [8].

The primary task of query processing is to find a strategy for executing each query over the network in the most efficient way. Query processing takes into consideration the distribution of the data, the communication cost, and the lack of sufficient locally available information. Query optimization refers to the process of ensuring that either the total cost or the total response time for a query is minimized. The choices to be made by a query optimizer sub-module of the query processor include: the order of executing relational operations; the access methods for the relating relations; the algorithms for carrying out the relational algebra operations; the order of data movements

Received by the editors: April 4, 2014.

2010 *Mathematics Subject Classification.* 68P15.

1998 *CR Categories and Descriptors.* H.2.4 [**D**atabase Management]: Systems – Distributed Databases.

Key words and phrases. distributed databases, query optimization, data transfer cost.

between sites. A good measure of resource consumption is the total cost that will be incurred in processing the query. In a distributed database system, the total cost to be minimized usually includes: CPU, I/O and communication costs.

Query optimization and the selection of join order are critical to the performance of practical relational database management systems. A query optimizer selects among the many alternative query execution plans, the one with the least estimated execution cost, according to a given cost function. The objective functions of query optimization may take many different forms. One may try to find a query evaluation plan that optimizes the most relevant performance measures, such as the response time, CPU, I/O, and network time and efforts, memory or storage costs, resources usage. The complexity of query optimization is basically determined by the number of alternative query execution plans, which grows exponentially with the number of relations involved in the query. Consequently, enumerative optimization strategies are prohibitively expensive and therefore unacceptable as the query sizes grow. Moreover, a database management system usually supports a variety of join algorithms for processing joins and a variety of indices for accessing individual relations, which increase the complexity. All query optimization algorithms primarily deal with the join queries.

Let's consider a distributed database with data and processing on m nodes $S = \{s_i | i = \overline{1,m}\}$. On each node of the system there is a subset of the data that is subject to query and update operations. The database is composed of a set of n fragments $F = \{f_j | j = \overline{1,n}\}$. A given fragment $f \in F$ has a size $dim(f)$ (given in bytes, pages, etc). Let $S(f)$ be the nodes where fragment $f \in F$ is stored and $F(s)$ the fragments stored on node $s \in S$.

A set Q of operations needs to be executed against the database in a given time interval. We assume for simplicity that most operations are read-queries. In order to execute a query $q \in Q$, the system generates an execution plan that can be represented as a tree. The leaf nodes are actual fragments of the distributed database, while internal nodes correspond to relational operators (unary or binary) to be evaluated.

A query execution needs data access to different fragments and intermediary results obtained by evaluating relational operators of the query. These intermediary results need to be transferred between processing nodes of the system. The fragment allocation order could have a large impact on the query execution time. Various fragment allocation methods are described in [2, 4, 6, 9, 12, 14, 15, 16, 17, 19].

During each query execution the system gathers various statistical information about the evaluation of relational operators in the execution plans. Using statistics and knowledge about data distribution, one could generally

propose an optimal query execution plan that minimizes the cost of inter-node data transfer.

The remaining of this paper is organized as follows. Section 2 presents relevant research on this subject, Section 3 describes the various information an evaluation engine could obtain while evaluating execution plans. In section 4 we provide an algorithm for finding the optimal node where each operator needs to be evaluated in order to minimize the cost of inter-node data transfer, for a given fragment allocation. In the last section we present the conclusions and future developments.

2. Related work

Many algorithms have been proposed for various aspects of query optimization, fragmentation, data and operation allocation. These algorithms may be divided into three major categories: deterministic search algorithms, randomized algorithms and genetic algorithms. The most prevalent technique in first category is dynamic programming, as used in System R [10]. The algorithm exhaustively searches through all plans for the query and prunes away bad sub-plans as early as possible. This algorithm is not time efficient for large number of joins and relations as it is exponential. There are many variations of this classical algorithm. Randomized algorithms make random choices as they walk thru the state space to find a local minima. The most successful of these algorithms called Two Phase Optimization [7] combines iterative improvement (a variant of hill climbing) with simulated annealing. The query optimization problem is a difficult combinatorial optimization problem with complicated objective functions. Genetic algorithms may be very effectively applied to search for solutions in the query optimization problem with a very large number of relations, see [21], [2].

Apers has discussed in detail the data allocation problem and their fragmentation in [1]. An heuristic algorithm for redistributing the fragments is proposed in [16]. The algorithm minimizes the size of the data transferred for solving a request. Assuming that a distribution of the fragments in the nodes of a network is known, the algorithm generates a plan to transfer data fragments, plan that will be used to evaluate a request. Other fragment allocation methods are described in [2, 4, 6, 9, 12, 14, 15, 16, 19].

3. Query Evaluation

Before actual execution and data retrieval for a given query q, the system generates a query execution plan. The execution plan decomposes q in a set of sub-queries, each sub-query corresponding to an evaluation algorithm for an operator from the relational algebra. Generally, and in our approach, the

execution plan is represented as a tree where leaf nodes are fragments and internal nodes are relational operators. Some of these operators need a single argument (unary - projection, selection), while others need two arguments (binary - join). The operator arguments in various phases of the execution are either fragments or the results of evaluating some other operators.

For a given query q we can determine a cost for the evaluation of all its sub-queries, a cost for transferring the required data between the nodes where sub-queries are executed and a cost for transferring the intermediate results from the where the root operator from the query execution plan is evaluated to the node where the final result of q is needed [5].

Let c_{ij} be the cost for transferring a data unit from a given node s_i to a different node s_j. When transferring an entire fragment, table or intermediary result T from s_i to s_j the actual transfer cost is given by $c_{ij} \times dim(T)$. We can assume without any over-simplifying that the transfer costs are constant $c_{ij} = 1$ if $i \neq j, c_{ii} = 0, 1 \leq i, j \leq m$.

The execution of a query q implies the postorder traversal of the associated execution plan (tree) and the evaluation of the relational operators from the internal nodes. Each operator is evaluated on a node of the system and implies a potential data transfer from other nodes of the system where its arguments are evaluated. After the operator in an internal node is evaluated we obtain a result of a given size that is independent of the nodes that provide the argument data. In order to assess the computation of the data transfer cost when executing a query q, we associate a *tag* to each node of the execution plan $(node, dim, ct)$, where:

- **node** - is the node where the fragment is stored (if a leaf), or where the relational operator is evaluated (if an internal node);
- **dim** - the size if the fragment or intermediate results;
- **ct** - the cost of data transfer required to evaluated the current operator. It is obtained as the sum of data transfer cost for the arguments of the current node and the required costs for the evaluation of these arguments at their origin.

For a leaf node, corresponding to the fragment f, the *tag* is $(node, dim, cost)$, where dim is the fragment size and $ct = 0$ if $node \in S(f)$, or $ct = dim$ if $node \notin S(f)$.

In the following paragraphs we introduce the *tag* computation methods for a binary and unary operators.

Let Θ_1 be an unary relational operator. Figure 1 shows the computation values involved in the computation of the *tag* for the Θ_1 node.

Using the *tag* for node A we obtain for Θ_1:

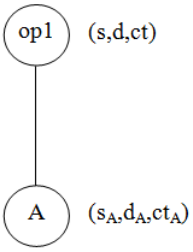


FIGURE 1. Unary operator tag

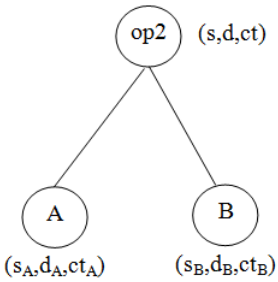


FIGURE 2. Binary operator tag

$$ct_{\Theta_1} = \begin{cases} ct_A & \text{if } s = s_A \\ ct_A + d_A & \text{if } s \neq s_A \end{cases}$$

The case of a binary operator is depicted in Figure 2:

if $s_A = s_B$ then: $ct_{\Theta_2} = \begin{cases} ct_A + ct_B, & \text{if } s = s_A \\ (ct_A + ct_B) + (a_A + d_B), & \text{if } s \neq s_A \end{cases}$

otherwise if $s_A \neq s_B$, then: $ct_{\Theta_2} = \begin{cases} (ct_A + ct_B) + d_B, & \text{if } s = s_A, \\ (ct_A + ct_B) + d_A, & \text{if } s = s_B, \\ (ct_A + ct_B) + (d_A + d_B), & \text{if } s \notin s_A, s_B \end{cases}$

A and B are the relations in our database fragmented horizontally as follows:

$$A = A_1 \cup A_2; \quad B = B_1 \cup B_2$$

Let us consider the following query:

$$q = A \times \sigma_C(B) \tag{1}$$

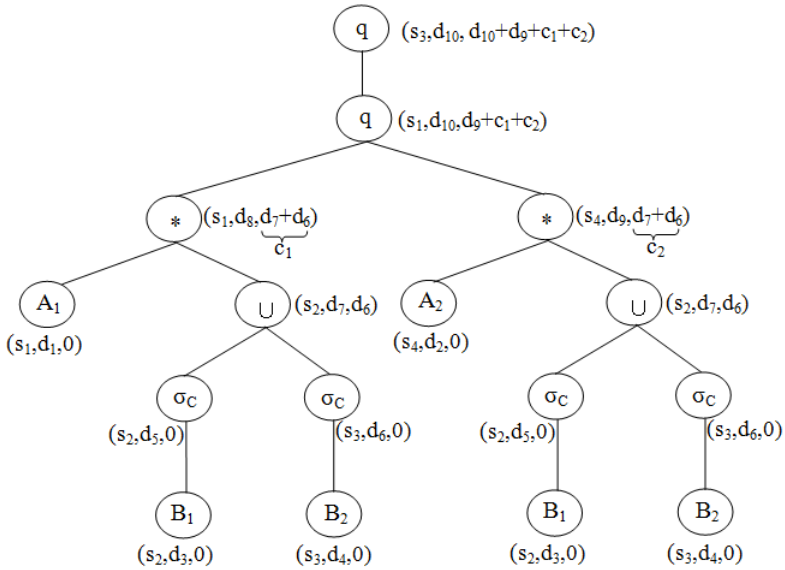


FIGURE 3. Node tags for query q.

where "$\times$" is the join operator and σ_C is a selection operator over C. This query can be transformed as follows:

$$\begin{aligned} q &= A \times \sigma_C(B) = \\ &= (A_1 \cup A_2) \times \sigma_C(B_1 \cup B_2) = \\ &= [A_1 \times \sigma_C(B_1 \cup B_2)] \cup [A_2 \times \sigma_C(B_1 \cup B_2)] = \\ &= [A_1 \times (\sigma_C(B_1) \cup \sigma_C(B_2))] \cup [A_2 \times (\sigma_C(B_1) \cup \sigma_C(B_2))] \end{aligned}$$

Suppose the database fragments are stored on four nodes:

$$\begin{aligned} &F(s_1) = \{A_1, B_1\}; F(s_2) = \{B_1\}; \\ &F(s_3) = \{B_2\}; F(s_4) = \{A_2\}. \end{aligned} \tag{2}$$

Supposing that the results for the query q needs to be returned on s_3, figure 3 presents the tags that appear in the nodes of the execution plan.

The figure also presents an additional node corresponding to the entire query q. Node tags (in the tree) correspond to a possible execution plan and allows to infer the data transfer costs. For each node of the execution plan a database node has been chosen for evaluating its operator.

We note here that the size of the fragments and intermediate results does not change if the operator evaluation is done on different nodes of the database system. In the next section we propose a method for finding the database node (station) where each operator needs to be evaluated such that the total data transfer cost be minimal.

4. Finding the query evaluation plan with minimal data transfer cost

For each node in the query evaluation tree, with its attached tags (like in figure 3) we add an additional root node corresponding for the whole query result and we determine the following values:

- $\underline{d}$ - size of data associated with a node;
- A vector $\underline{c} = [c_1, \ldots, c_m]$, where m is the number of stations composing the distributed database. A component of the vector c_i has the following meaning:
  - for a leaf node,corresponding to a fragment f, is the cost of data transfer for the given fragment to station (node) s_i;
  - for an internal node, corresponding to a relational operator, is the *minimum data transfer cost* if the evaluation of this operator takes place on s_i;
- $\underline{sr}$ - a database station where is recommended to evaluate the current operator (for an internal node of the execution plan), or where the fragment needs to be read (for a terminal node).

In the first step of our method we compute the vector $\underline{c}$ for each node of the query evaluation plan using a post-order traversal order. For a given current node, the computation of the $\underline{c}$ uses the values of $\underline{d}$ and $\underline{c}$ associated with the operands nodes. If the current node is a leaf node, corresponding to a fragment f, then: $c_i = \begin{cases} 0, \text{ if } s_i \in S(f), \\ dim(f), \text{otherwise.} \end{cases}$

The second step starts from the root of the query evaluation plan. The network station where the response of the query q needs to be returned will be the $\underline{sr}$ value associated to this node. For a given current node, starting from the root, we will compute the values $\underline{sr}$ for the associated operand nodes.

There are two ways for computing the values of the $\underline{c}$ vector for the current node depending if the associated operator is binary or unary.

Let Θ_1 be an unary operator. The c_i values $i = \overline{1, m}$ are the minimum data transfer costs if Θ_1 is evaluated on station s_i. Figure 4 shows the construction of the $\underline{c}$ vector for an unary operator. If the argument (operand) A is evaluated (or stored) in a station $s_j, i \neq j$, then overall cost of evaluating the node A on s_j is incremented with d_A, the cost of data transfer from station s_j to station

s_i. The total transfer cost for A becomes thus $c_j^A + d_A$. If $i = j$, then the cost for the node A is c_i^A. In the general case, the formula for a node is given by:

$$c_i = \min\left\{c_1^A + d_A, \ldots, c_{i-1}^A + d_A, c_i^A, c_{i+1}^A + d_A, \ldots, c_m^A + d_A\right\} \tag{3}$$

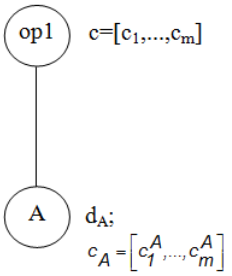


FIGURE 4. Cost vector attached to an unary operator node.

For a binary operator we use a similar computation method as for the unary operator. Figure 5 shows the intervening values and their computation is explained bellow.

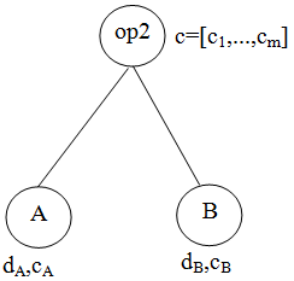


FIGURE 5. Cost vector attached to a binary operator node.

The value of the $c_i, i = \overline{1,m}$ components in the case of a binary operator Θ_2 are the minimum transfer costs if the operator is evaluated on station s_i. They are obtained as follows:

$$\begin{aligned} c_i = &\min\left\{c_1^A + d_A, \ldots, c_{i-1}^A + d_A, c_i^A, c_{i+1}^A + d_A, \ldots, c_m^A + d_A\right\} + \\ &+ \min\left\{c_1^B + d_B, \ldots, c_{i-1}^B + d_B, c_i^B, c_{i+1}^B + d_B, \ldots, c_m^B + d_B\right\} \end{aligned} \tag{4}$$

In the following we assume that the value of the $\underline{\mathrm{d}}$ and the components of the $\underline{\mathrm{c}}$ vector have been computed for a given query q on all nodes of the query

evaluation plan (tree). The value for d is usually obtained from statistics and estimations stored in the database dictionary.

Let c_q be the vector associated with query q, represented in the evaluation plan as the root node (or node q). The value sr associated to node q is given by the station sq where we need to return the result of the query execution, so $sr_q = s_q$. The value c^q_{sq} from the $c_q = \left[c^1_1, \ldots, c^q_m\right]$ vector is the minimum data transfer cost if q is evaluated on station sq. This value is computed according to equation 3 using the operand node of q like depicted in figure 4.

We start with this initial value for the root node and sr. We traverse the nodes of the evaluation tree from the root node to the leafs (this can be a pre-order traversal) and compute the values for sr for each operand (argument) of a given current node.

For an unary operator Θ_1, as in figure 4, the values for c and sr are known. This operator needs to be evaluated on station s_{sr}. The c_{sr} value is the minimum data transfer cost if Θ_1 is evaluated on station s_{sr} and is computed according to equation 3. The minimum value (according to equation 3) could be obtained on multiple stations. For simplicity, in the case we obtain the minimum value on multiple stations we choose one (any one) station that achieves the minimum. Let this station be sr_A - the station attached to node A. Figure 6 show a numerical example for c computation on unary operators.

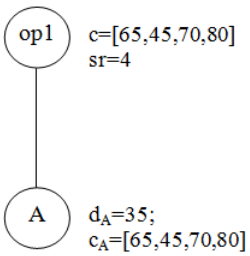


FIGURE 6. Example: Cost vector computed for an unary operator.

$sr = 4 \Rightarrow c_4 = 80 = \min\{65+35, \underline{45+35}, 70+35, \underline{80}\} \Rightarrow sr_A \in \{2,4\}$. Stations s_2 and s_4 are the stations that incur the lowest data transfer cost when evaluating Θ_1.

For a binary operator node, like in figure 5, we can compute the values of sr_A and sr_B using the Θ_2's sr value and sizes of operands A and B. Figure 7 shows a numerical example for computing these values.

$$\begin{aligned} sr = 4 \rightarrow d_4 = 80 \quad &= \quad \min\{50+30, 25+30, 40+30, \underline{30}\} + \\ &+ \quad \min\{\underline{15+35}, 20+35, 30+35, \underline{50}\}. \end{aligned}$$

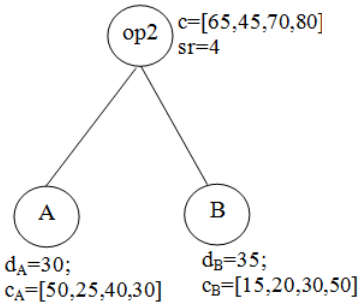


FIGURE 7. Example: Cost vector computed for a binary operator.

We obtain thus: $sr_A = 4, sr_B \in \{1, 4\}$.

5. Experiments and numerical test

In the following, we are going to compute the tags associated with the evaluation plan for the query expressed in equation 1, on the evaluation tree presented in figure 3. Using actual values for fragment sizes and intermediate results, we traverse in the first step the tree in post-order, from the leafs to the root in order to compute the $\underline{c}$ values on each node. The second traversal of the evaluation tree, from the root to the leafs (pre-order) is used to compute the $\underline{sr}$ values, and thus the stations where each query operator needs to be evaluated in order to minimize the data transfer cost. The obtained results are depicted in figure 8.

The $\underline{c}$ values for the leaf nodes (fragments) are computed from the the information on the fragment allocation to database stations given by the equation 2.

6. Conclusions and future work

In this paper we propose a cost based optimization of a query in a distributed relational database, by applying some metrics to its evaluation tree. We start from the general assumption that the evaluation plan is represented as a tree with the leaf nodes storing the actual database fragments and the internal nodes representing operators of the query. Once a candidate evaluation plan is determined, our method tries to minimize the cost of data transfers when executing the query against the real database by dynamically computing the stations of the system that are optimal for each query operator evaluation. We also take in account the data transfer to the station where the results needs to be obtained. The proposed method mathematically chooses the optimal node for each query operator evaluation, based on some usual statistical

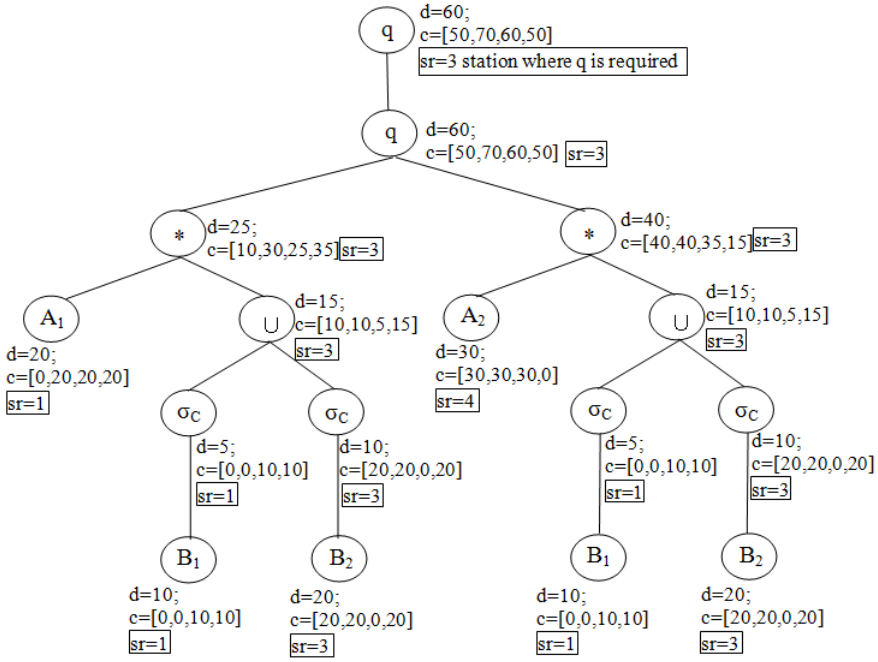


FIGURE 8. Evaluation of the query in equation 1

database information and by attaching *tags* to operator nodes such that overall data transfer cost is minimal. The algorithm uses two traversals of the query evaluation tree in order to extract the *operator-to-station* execution affinity. We aim to use the results of the proposed method to propose dynamic re-fragmentation of the database or data replications in order to minimize data transfer and processing costs in distributed databases.

REFERENCES

[1] P.M.G. Apers, *Data Allocation in Distributed Database Systems*, ACM Trans. on Database Systems, 13(3), (1988), pp. 263-304.

[2] C.H. Cheng, W.K. Lee, K.F. Wong, *A Genetic Algorithm-Based Clustering Approach for Database Partitioning*, IEEE Transactions on Systems Man and Cybernetics Part C-Applications and Reviews, 32, (2002), pp. 215-230.

[3] R. Diestel, *Graph Theory*, Springer-Verlag, Heidelberg 2000, Electronic Edition.

[4] J. Graham, *Efficient Allocation in Distributed Object Oriented Databases*, Proceedings of the ISCA 16th International Conference on parallel and Distributed Computing Systems, Reno Nevada, August (2003), pp. 407-412.

[5] G. Graefe, *Query Evalution Techniques for a Large Database*, ACM Computing Surveys, 25, (1993) pp. 73-90.

[6] Y. Huang, J. Chen, *Fragment Allocation in Distributed Database Design*, Journal Of Information Science And Engineering, 17, (2001) pp. 491-506 .
[7] Yannis E. Ioannidis, Youngkyung Cha Kang, *Randomized Algorithm for Optimizing Large Join Queries*. Procedings of ACM SIGMOD International Conference on Management Of Data, (1990), pp. 312-321 .
[8] D. Kossmann, *The State of the Art in Distributed Query Processing*, ACM Computing Surveys, Vol. 32, Issue 4, (2000) pp. 422-469.
[9] S.T. March, S. Rho, *Allocating Data and Operations to nodes in a distributed database design*, IEEE Trans. on Knowledge and Data Engineering, Vol. 7 (2), (1995), pp. 305-317.
[10] M. T. Ozsu, P. Valduriez, *Principles of Distributed Database Systems*, 2nd ed., Prentice-Hall International Editions, 1999.
[11] M. S. Sacco, S. B. Yao: *Query Optimization in Distributed Database Systems*, Advances in Computers, Vol. 21, Academic Press, New York, (1982), pp. 225-273.
[12] Rajinder Singh, Gurvinder Singh, Varinder Pannu, *Optimized Access Strategies for a Distributed Database Design*, International Journal of Data Engineering (IJDE), Vol. 2, Issue 3, (2011), pp. 102-110, .
[13] Rajinder Singh Virk, Gurvinder Singh, *Optimizing Access Strategies for a Distributed Database using Genetic Fragmentation*, International Journal of Computer Science and Network Security, VOL.11 No.6,(2011), pp. 180-183.
[14] A. Sleit, W. AlMobaideen, S. Al-Areqi, A. Yahya, *A Dynamic Object Fragmentation and Replication Algorithm in Distributed Database Systems*, American Journal of Applied Sciences 4 (8), (2007) pp. 613-618.
[15] L. Tambulea, M. Horvat, *Dynamic Distribution Model in Distributed Database*, Int. J. of Computers, Communications & Control, Vol. III (2008), Suppl. issue: Proceedings of ICCCC (2008), pp. 512-515.
[16] L. Tambulea, M. Horvat, *Redistributing fragments into a distributed database*, Int. J. of Computers, Communications & Control, Vol. III (2008), No. 4, pp. 384-394.
[17] T. Ulus, M. Uysal, *Heuristic Approach to Dynamic Data Allocation in Distributed Database Systems*, Pakistan Journal of Information and Technology 2 (3),(2003) pp. 231-239.
[18] S. Upadhyaya, S. Lata, *Task allocation in Distributed computing VS distributed database systems: A Comparative study*, IJCSNS International Journal of Computer Science and Network Security, VOL.8 No.3, (2008), pp. 338-346.
[19] O. Wolfson, S. Jajodia, *An Algorithm for Dynamic Data Distribution*, Proceedings of the 2nd Workshop on the Management of Replicated Data (WMRD-II), Monterey, CA, (1992), pp. 62-65.
[20] C.T. Yu, C.C. Chang, *Distributed Query Processing*, ACM Computing Surveys, Vol. 16, (1984), pp. 399-433.
[21] Zehai Zhou, *Using Heuristics and Genetic Algorithms for Large Scale Database Query Optimization*, Journal of Information and Computing Science Vol. 2, No. 4, (2007), pp. 261-280

Department of Computer Science, Faculty of Mathematics and Computer Science, Babeş-Bolyai University, Kogălniceanu 1, 400084 Cluj-Napoca, Romania
E-mail address: leon@cs.ubbcluj.ro
E-mail address: dadi@cs.ubbcluj.ro
E-mail address: ivarga@cs.ubbcluj.ro

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LIX**, Number 1, 2014

A STUDY REGARDING INTER DOMAIN LINKED DOCUMENTS SIMILARITY AND THEIR CONSEQUENT BOUNCE RATE

DIANA HALIŢĂ AND DARIUS BUFNEA

Abstract. Then main objective of linking inter domain documents is to offer to the reader access to supplementary, semantic related information. However, linking web domains is sometimes artificially used, especially when the goal is to abusively increase the page rank of the destination domain. This paper presents a study regarding inter domain linked documents similarity and their consequent bounce rate. For that, we have advanced a series of experiments which outlines how similarity functions' behavior correlates with a website bounce rate. The method presented here could be used to identify within a web site improper placed outgoing links such as ads or spam links. Based on that, a search engine could fine the results in SERP by downgrading any website that fall in the above presented category.

1. Introduction

In a certain document context, one of the goals of referring a document within another one is to give to the reader access to more semantic related information to the information being currently read. Two common examples in this direction are a scientific article citing one of its references or a web page linking naturally to another web page.

When it comes to linking inter domain web pages (i.e. HTML links that points to documents hosted on a different domain), often the linking is performed in an abusive manner in order to artificially increase the page rank of the destination document or domain and not to lead the visitor to a more semantic related information to the one he is currently interested in.

Received by the editors: April 18, 2014.

2010 *Mathematics Subject Classification.* 68U15, 68M11, 68U35.

1998 *CR Categories and Descriptors.* H.5.3 [**Group and Organization Interfaces**] - Web-based interaction; H.5.4 [**Hypertext/Hypermedia**] - Navigation; I.7.5 [**Document Capture**] - Document analysis.

Key words and phrases. bounce rate, document similarity, page ranking, identify improper placed links.

In the majority of cases, such abusive links are either site wide (for example ads), these one being the most easy to detect, or it might be automatically placed inside the absolute content of a web page (i.e. the article/post's effective content) by different advertising modules integrated in the Content Management System of the source web site.

This paper analyzes the possible relation between the similarity of the source and the external linked document and the bounce rate generated within the destination domain by this link. Such a possible relation could lead to bounce rate estimation and disclosure for any external link between any two third party web sites. A high estimated bounce rate for the destination site via such a link may indicate an improper link, i.e. an abusive, spam or ad link.

2. Analysis of bounce rate in relation with inter linked documents' similarity

Bounce rate represents the percentage of the visitors who enter a site and leave it rather than continue visiting other pages. Generally this have two meanings. First of all, one may find the exactly desired information. For example, a user search the definition of bounce rate and this information is provided accurately by the first web page in SERP (Search Engine Results Page). He is satisfied with the provided definition and he leaves the site without accessing any other result page. However, in most cases the user may not be satisfied with information shown on a particular web page in SERP and he may leave it immediately in order to access the next one returned in SERP. It is widely accepted, that a greater bounce rate, mainly due to the second case, means something negative and that value is directly associated with the quality of the content.

The higher similarity between the outgoing web page and the external referred page is obtained, the more similar content is provided to the user. This leads to a smaller bounce rate for the destination domain, the visitor having much more to read about what he is interested in.

Example:

- An Internet based forum dedicated to pets lover contains a topic with a link that points to a dog raising website. Such a link offers a plus of information to a visitor being susceptible of generating a smaller bounce rate for the destination web site.
- The admission web site of our university contains basic information about the admission process. More detailed information about the admission process is available via a web link on each faculty's web site

which is hosted on a different domain. Such a link will also offer additional semantic related information to the reader as the information he is interested in, generating a smaller bounce rate for a certain faculty's web site.

Counterexample:

- An abusive link, such as a spam or an ad link, in many cases points to a site / external page having a content that is not semantic related with the source. Visitors that follow such a link will consequently generate a higher bounce rate for the destination site: i.e. will click the link, access the destination web page and then close it or return back to the previously accessed web page. In fact, targeted advertising was introduced in order to increase conversion from the advertised site point of view [3].

In order to analyze the possible connection between linked documents similarity and their consequently generated bounce rate we will test in the respect of this some basic similarity functions such as: Cosine, Jaccard, Sorensen and Jaro-Winkler. In a working paper, we will also address the same topic using semantic similarities.

3. Previous Work

As the Internet have evolved and surfing the Web, as its main application, become a common and a daily based activity for the society, the spam linking spread as a negative phenomenon that affect mainly the quality of search results return by a search engine. The biggest companies involved in the search industry and equally the research community have step up their efforts in order to identify solutions to detect and limit this negative phenomenon's impact.

Ever since link analysis was used in building search engines optimization algorithms, corresponding spamming techniques have been developed [9]. As a consequence, multiple negative effects were caused by spamdexing. This effects lead to the appearance of new challenges in this area research. In a comprehensive survey on the main principles on web spam detection [8], authors had categorize all existing techniques into three categories based on the type of information they use: content-based methods, link-based methods, and methods based on non-traditional data analysis such as user behavior, clicks and HTTP sessions. These techniques are able to detect up to 80% of spam pages [2] and they should be applied together, at least by combining link based and content based analysis [1]. Previous studies related to web spam detection were using automatic supervised or unsupervised classification [1], power-law distribution, algorithms for collusion detection [4] or confusion

matrix and precision-recall matrix [7]. That was a key challenge for search engine industry, so spamdexing was cast into a machine learning problem of classification on directed graphs [9].

4. Predicting bounce rate using external linked documents' similarity

4.1. **Ideal similarity.** Based on the above observations, that high content related web pages are less susceptible for generating a high bounce rate and poor content related pages can lead to a higher percent of visitors that bounce, we intend in this section to study the possible correlation between bounce rate and several similarities functions. In the case of an ideal similarity function, depicted in figure 1, the bounce rate will follow a linear regression path.

By using a properly chosen similarity function, one could estimate the bounce rate of a third party external link based only on content of linked documents' similarity (currently, bounce rate may only be obtain by a site's owner using web analytics tools such a Google Analytics). Such a method could be used by a third party (for e.g. the search engine) to identify improper placed and abusive links such as ads or spam links. Consequently, a web site that relies on a large number of links that fall in the above categories could be fined by a search engine.

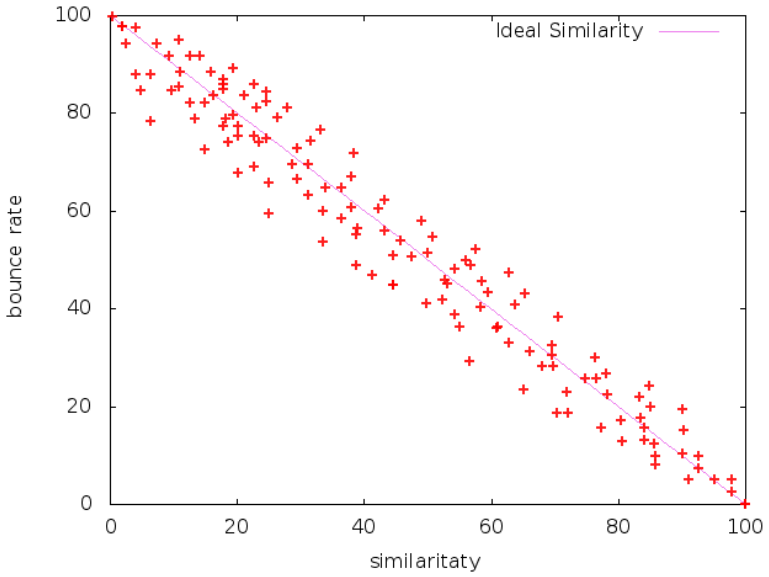


Figure 1. Ideal similarity

The perfect correlation between bounce rate and similarity of inter domain linked documents, shown in an ideal case [fig:1], is not always found, especially considering usual mathematical similarity functions. We choose to take into consideration more similarity functions in order to determine which one fits better our goal. Intuitively, we say that a similarity function is better than other if the pairs (similarity, bounce rate) obtained from each link between two inter domain linked pages are closer to the diagonal (as we can see in the an ideal case [fig:1]). In our case, similarity represents the horizontal coordinate (abscissa) of the point in a two-dimensional rectangular Cartesian coordinate system and bounce rate represents the vertical coordinate (ordinate) of the above considered system. Taking into consideration the line which is parallel with the secondary bisector of the Cartesian coordinate system and passes through the points $(100, 0)$ and $(0, 100)$, a similarity function is the best if the sum of all distances, from the graphical representation points, represented by the above named pairs, to that line is minimal, i.e. if

$$\sum \frac{|x_i + y_i - 100|}{\sqrt{2}}$$

is minimal, where x_i and y_i are the coordinates of a point on the graphical representation.

4.2. **Experimental results.** Various similarity functions can be used for determining whether two strings are similar. In this paper, we have tested all gathered data against the following similarity functions: Cosine Similarity, Jaro-Winkler Similarity, Sorensen Similarity and Jaccard Similarity. The Cosine Similarity measures the angle between two vectors, but it does not take into consideration the order of the strings. Jaro-Winkler similarity is a lot more accurate especially because it takes into consideration the order of the strings, but it is designed and best suited for short strings. The Jaccard coefficient is defined as the length of the intersection divided by the length of the union of the two sets of strings. An important class of problems, that Jaccard similarity addresses well, is finding textually similar documents in a large corpus, such as the Web. Sorensen similarity coefficient is similar to Jaccard coefficient, but it has some different properties, including retaining sensitivity in more heterogeneous data sets and gives less weight to outliers.

In order to have the best possible accuracy of the results and for reducing the noise induced in the similarity algorithm by the master pages' HTML code, we have tested all the similarity functions for the absolute content of the documents (i.e. we ignore all the content that falls within the footer, header or menu).

The absolute content of a web page was completely determined using a Java library named *boilerpipe*, which is able to remove or extract full text from HTML pages. This library provides algorithms which detects and removes the master page template around the main content of a web page. This library was released under Apache License 2.0.

In order to follow our ideas we took for experimental purpose, an educational website $http://www.cs.ubbcluj.ro$ (Computer Science Faculty website), and we generated all the triplets $(landingpage, refferer, bounce\ rate)$ through two methods:

- a client side tool provided by Google, named Google Analytics;
- a server side tool, integrated in the website's master page template, developed by us.

We have chosen to experiment on the above website because we have administrator rights on the faculty website and we may properly measure for each external incoming link the generated bounce rate.

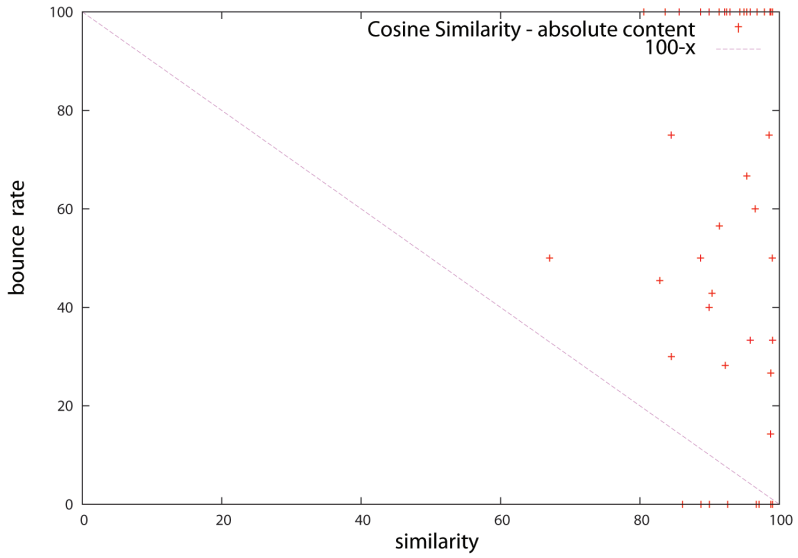


FIGURE 2. Cosine similarity

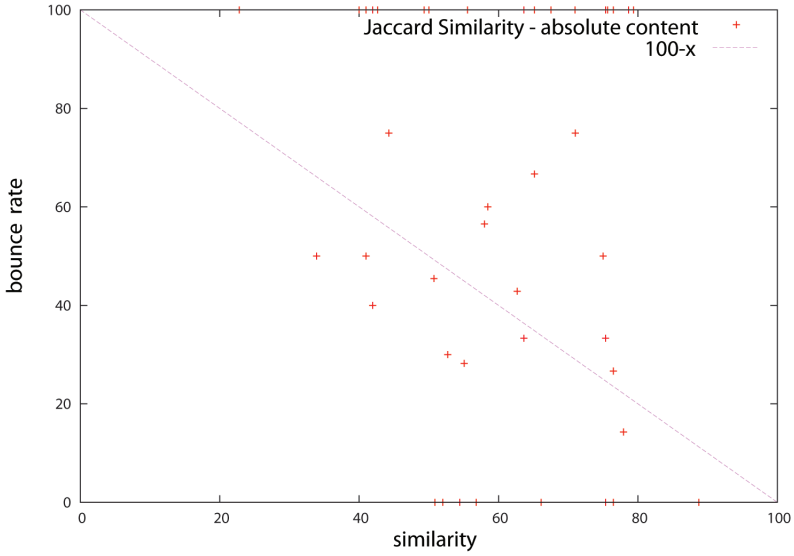


FIGURE 3. Jaccard Similarity

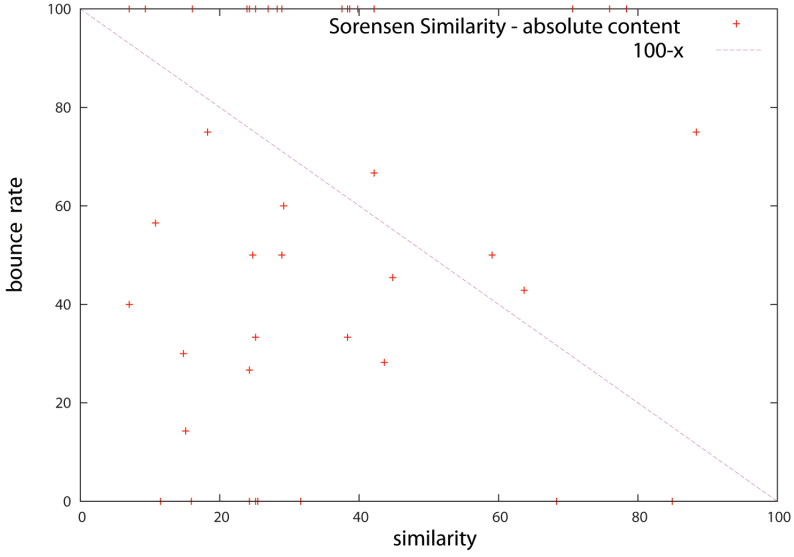


FIGURE 4. Sorensen similarity

As we can see in the above figures, the best similarity function which gives us the expected results is Jaccard distance applied only to the absolute

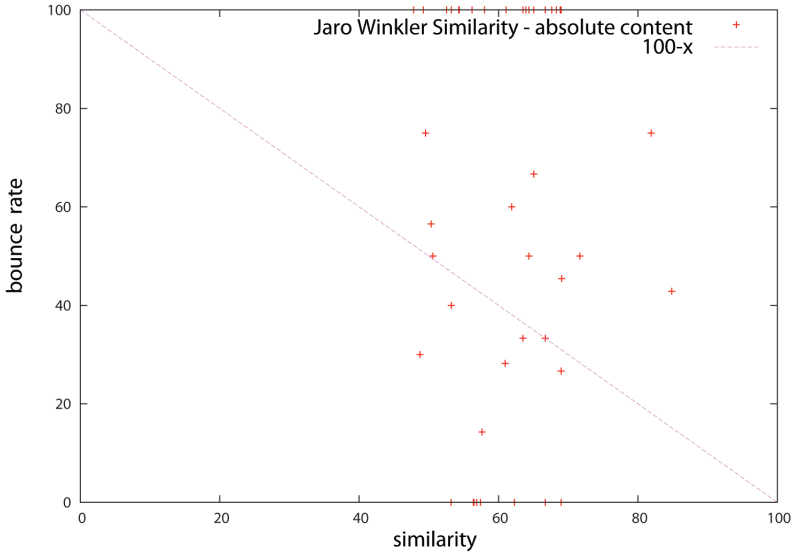


FIGURE 5. Jaro-Winkler Similarity

contents of the web pages. Given this, we will use this similarity function for our future graphical representations.

We also can prove that Jaccard is the best similarity function in our case, by calculating for each similarity function used and for both absolute content and full content comparison, the distance from all points which are represented on the figure to the line of equation $y = x - 100$.

Similarity function	Method	Value of the Sum
Cosine	absolute content	1804.476000579978
Jaccard	**absolute content**	**1414.03085464388**
Sorensen	absolute content	1769.3699189543242
Jaro-Winkler	absolute content	1528.5359097516346

TABLE 1. Sum of the distances from all points to the line of equation $y = x - 100$

5. Conclusions and Future work

In this paper we have advanced a series of experiments in order to see whether the bounce rate correlates with different similarity functions. A high content similarity between linked and source content may imply future visitor

requests for pages hosted on the destination domain, these action reducing the bounce rate implied by such links. Starting with this idea it is possible, as future work, to obtain better results if we analyze the similarity of the referrer's page content with the content of each internal link found in the corresponding landing page.

Future work may also imply testing this ideas on semantic similarity functions. Other ideas would be giving different weights to similarity functions or choosing a similarity function and weighting and fine tuning various specific properties of the content, such as URL, page headings, page title or keywords.

A work in progress paper of the same authors is studying scrapper sites identification based on their content similarity with the content they automatically retrieved and usually link to.

References

[1] L. Becchetti, C. Castillo, D. Donato, *Link-Based Characterization and Detection of Web Spam*, 2nd International Workshop on Adversarial Information Retrieval on the Web, AIRWeb, Seattle, USA, August 2006, pp. 1-8

[2] L. Becchetti, C. Castillo, D. Donato, S. Leonardi,R. Baeza-Yates, *Link Analysis for Web Spam Detection: Link-based and Content Based Techniques*, ACM Transactions on the Web (TWEB), Volume 2, Issue 1, New York, USA, February 2008, pp. 1-41

[3] A. Farahat, M. Bailey, *How Effective is Targeted Advertising?*, Proceedings of the 21st World Wide Web Conference 2012, Lyon, France, April 16-20, 2012, pp. 111-120

[4] Z. Gyongy, H. Garcia-Molina, P. Berkhin, J. Pedersen, *Link Spam Detection Based on Mass Estimation*, 32nd International Conference in Very Large Data Bases (VLDB), Seoul, Korea, 2006, pp. 439-450

[5] A. Huang, *Similarity Measures for Text Document Clustering*, Proceedings od the New Zealand Computer Science Research Student Conference, Hamilton, New Zealand, 2008, pp. 49-56

[6] J. Leskovec, A. Rajaraman, J. Ullman, *Mining of Massive Datasets*, Cambridge University Press, 2010

[7] M. Najork, *Detecting Spam Web Pages through Content Analysis*, International World Wide Web Conference Comittee, Edinburgh, Scotland, 2006, pp. 83-92

[8] N. Spirin, J. Han, *Survey on Web Spam Detection: Principles and Algorithms*, ACM SIGKDD Explorations Newsletter, Volume 13, Issue 2, December 2011, pp. 50-64

[9] D. Zhou, C. Burges, T. Tao, *Transductive Link Spam Detection*, Proceedings of the 3rd International Workshop on Adversarial Information Retrieval on the Web, AIRWeb, New York, USA, ACM Press, 2007, pp. 21-28

Babeş-Bolyai University, Faculty of Mathematics and Computer Science, 1 M. Kogălniceanu St., 400084 Cluj-Napoca, Romania

E-mail address: diana.halita@ubbcluj.ro, bufny@cs.ubbcluj.ro

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LIX**, Number 1, 2014

MULTIOBJECTIVE APPROACH OF MULTI-DIMENSIONAL TIME SERIES CLUSTERING

RAMONA STOICA

Abstract. The multidimensional time series are a generalization of the single time series and are more difficult to cluster due to the higher number of parameters used to characterize a data instance. In this work we formulate the multidimensional time series clustering problem as a multiobjective problem and implement several distance measures in the k-means clustering algorithm in order to see the effect of the similarity measure in the clustering process. All the measures are geometrical distances. We used four data sets in order to validate the results. The Euclidean distance which is the most used one does not seem to be the most adequate measure in multidimensional clustering.

1. Introduction

Time series data is a sequence of real numbers that represent the measurements of a real variable at equal time intervals. A data stream is an ordered sequence of points $x_1, \ldots, x_n$. These data can be read or accessed only once or a small number of times. A time series is a sequence of real numbers, each number indicating a value at a time point. Data flows continuously from a data stream at high speed, producing more examples over time in recent real world applications. Most of the time series encountered in cluster analysis are discrete time series. When a variable is defined at all points in time the time series is continuous. Clustering of time series data has applications in an extensive assortment of fields and has attracted a large amount of research [1, 2, 3, 4, 5, 6, 7]. Multidimensional time series are an extension and generalization of regular time series. They have more impact nowadays as most of the data consists of more parameters which are measured over time and decision has to be made considering the behavior of all these parameters together. We

Received by the editors: January 27, 2014.

2010 *Mathematics Subject Classification.* 68P15, 68T05.

1998 *CR Categories and Descriptors.* I.2.6[**Computing Methodologies**]: Artificial Intelligence – *Learning*; I.2.8[**Computing Methodologies**]: Problem Solving, Control Methods, and Search – *Heuristic methods*.

Key words and phrases. Bioinformatics, Dynamic clustering.

propose to investigate in this paper the behavior of k-means algorithm for several multidimensional time series data. We compare versions of k-means for several distance measures. The paper is organized as follows: Section 2 introduces the clustering problems, some similarity distances and some approaches. Section 3 describes the multidimensional time series data, Section 4 presents the k-means for multidimensional time series data clustering and the distance measures that we used, Section 5 contains experiments and comparisons and Section 6 presents the conclusions of this work.

2. Clustering: basic notions

Clustering refears to grouping together data samples that are similar in some way, according to some criteria. It is a form of unsupervised learning because there are no examples showing how the data should be grouped together.

A *cluster* is a collection of data objects that are:

- similar to one another within the same cluster
- dissimilar to the objects in the other clusters.

There are several ways to define similarity and dissimilarity between clusters. These definitions depend on:

- the type of the data considered
- what kind of similarity we are looking for.

Similarity and dissimilarity between objects is often expressed in terms of a distance measure $d(x, y)$. Ideally, every distance measure should be a metric, i.e., it should satisfy the following conditions [8]:

(1) $d(x, y) \geq 0$
(2) $d(x, y) = 0$ *iff* $x = y$
(3) $d(x, y) = d(y, x)$
(4) $d(x, z) \leq d(x, y) + d(y, z)$

2.1. Similarity and dissimilarity measures.

In this section we briefly review the concepts of similarity and dissimilarity in the context of clustering and we present the similarity measures used later on in the paper.

2.1.1. *Similarity.* The *similarity measure* indicates the strength of the relationship between two data points. The more the two data points resemble one another, the larger the similarity coefficient is [1]. A *metric* is a distance function f that satisfies the following four properties [9]:

(1) non negativity: $f(x, y) \geq 0$
(2) reflexivity: $f(x, y) = 0 \Leftrightarrow x = y$
(3) commutativity: $f(x, y) = f(y, x)$

(4) triangle inequality: $f(x, y) \leq f(x, z) + f(y, z)$

where x, y, and z are arbitrary data points.

Several similarity distances exist. We present some of them which are further used in our implementation.

Euclidean distance. For two data points $\mathbf{x}$ and $\mathbf{y}$ in d-dimensional space, the Euclidean distance between them is defined by:

$$d_{euc}(x, y) = [\sum_{j=1}^{d}(x_j - y_j)^2]^{\frac{1}{2}} = [(x - y)(x - y)^T]^{\frac{1}{2}},$$

where x_j and y_j are the values of the j_th attribute of x and y, respectively. The squared Euclidean distance is defined as:

$$d_{euc}(x, y) = d_{euc}(x - y)^2 = \sum_{j=1}^{d}(x_j - y_j)^2 = (x - y)(x - y)^T$$

Manhattan distance. Manhattan distance is defined to be the sum of the distances of all attributes. That is, for two data points $\mathbf{x}$ and $\mathbf{y}$ in a d-dimensional space, the Manhattan distance between them is given by:

$$d(x, y) = \sum_{j=1}^{d}|x_j - y_j|$$

Maximum distance. Maximum distance is defined to be the maximum value of the distances of the attributes; that is, for two data points $\mathbf{x}$ and $\mathbf{y}$ in d-dimensional space, the maximum distance between them is given by:

$d_{max}(x, y) = max_{1<i<j<d}|x_j - y_j|$

Average distance. The average distance is derived from the Euclidean distance. Given two data points $\mathbf{x}$ and $\mathbf{y}$ in a d-dimensional space, the average distance is defined by:

$d_{ave}(x., y) = (\frac{1}{d}\sum_{j=1}^{d}(x_j - y_j)^2)^{\frac{1}{2}}$

2.2. **Clustering algorithms: k-means.** Clustering algorithms can be divided into two main classes:

(1) hierarchical algorithms: divide the data set into a sequence of partitions
(2) partitioning algorithms: divide the data set into a single partition

In this paper we deal with a variation of the k-mean clustering algorithm.

The k-means algorithm [10] is one of the most used clustering algorithms. It was designed to cluster numerical data in which each cluster has a center called the mean. The k-means algorithm is classified as a partitional or non-hierarchical clustering method [11]. In this algorithm, the number of clusters k is assumed to be fixed.

The algorithm has the following main steps:

(1) Pick a random number k of cluster centers

(2) Assign every item to its nearest cluster center using a similarity or distance measure (e.g. Euclidean distance)
(3) Move each cluster center to the mean of its assigned items
(4) Repeat steps 2 and 3 until change in cluster assignments is less than a threshold

There is an error function in this algorithm which, for given initial k clusters, allocates the remaining data to the nearest clusters and then repeatedly changes the membership of the clusters according to the error function until the error function does not change significantly or the membership of the clusters no longer changes. The k-means algorithm [12, 13, 8] is described below.

K-means algorithm
Require: Data set D, Number of Clusters k, Dimensions d:
{Ci is the i^{th} cluster}
{*1. Initialization Phase*}
1: ($C1$, $C2$,...,Ck) = Initial partition of D.
{*2. Iteration Phase*}
2: **repeat**
2.1: d_{ij} = distance between data i and cluster j;
2.2: n_i = arg $\min_{1 \leq j \leq k} d_{ij}$;
2.3: Assign case i to cluster n_i;
2.4: Recompute the cluster means of any changed clusters above;
3: **until** no further changes of cluster membership occur in a complete iteration
4: Output results.

The computational complexity of the algorithm is $O(nkd)$ per iteration [14, 8], where d is the dimension, k is the number of clusters, and n is the number of data points in the data set.

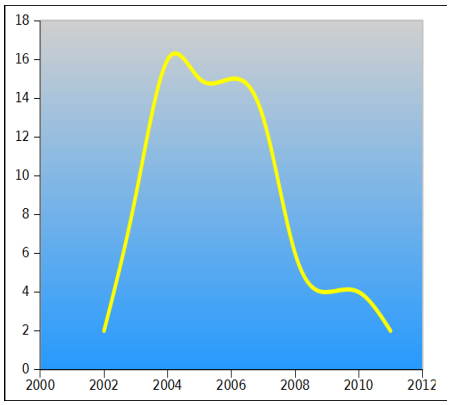
There are some drawbacks with the k-means algorithm:

- result can vary significantly depending on initial choice of seeds (both number and position);
- can get trapped in local minimum – it often terminates at a local optimum;
- to increase the chance of finding the global optimum: restart with different random seeds;
- must pick number of clusters before hand;
- all items are forced into a cluster;
- it is too sensitive to outliers;
- it does not perform well on high dimensional data;
- it only works with numerical data.

3. Time series and multidimensional time series clustering

Time series data is a sequence of real numbers that represent the measurements of a real variable at equal time intervals. Figure 1 shows an example of a time series that has years as unit time intervals.

Figure 1. A time series data example.



A data stream is an ordered sequence of points $x_1 \ldots x_n$. These data can be read or accessed only once or a small number of times. A time series is a sequence of real numbers, each number indicating a value at a time point. Data flows continuously from a data stream at high speed, producing more examples over time in recent real world applications.

Most of the time series encountered in cluster analysis are discrete time series. When a variable is defined at all points in time the time series is continuous. In general, a time series can be considered as a mixture of the following four components [15, 8]:

(1) a trend (the long-term movement);
(2) fluctuations about the trend of greater or less regularity;
(3) a seasonal component;
(4) a residual or random effect.

Clustering time series is a problem that has applications in an extensive assortment of fields and has recently attracted a large amount of research. Time series data are frequently large and may contain outliers. In addition, time series are a special type of data set where elements have a temporal ordering. Therefore clustering of such data stream is an important issue in the data mining process. Numerous techniques and clustering algorithms have been proposed earlier to assist clustering of time series data streams. The clustering algorithms and their effectiveness on various applications are compared to developing a new method to solve the existing problem.

Clustering of time series data has applications in an extensive assortment of fields and has attracted a large amount of research [16, 15, 17, 18, 19, 20, 21, 22].

3.1. **Multidimensional time series clustering.** A time series is defined as an array $X = (x_1, x_2, \ldots, x_n)$ of measurements in time for a given parameter (or variable).

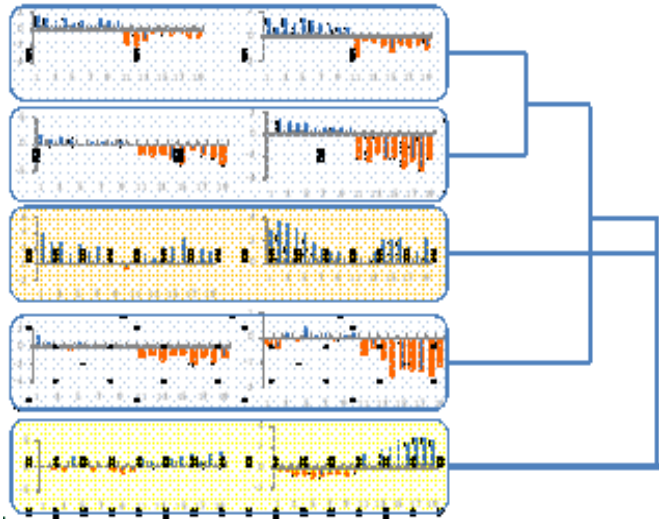
A *multidimensional time series* is defined as:

$$X = \begin{pmatrix} X_1 \\ X_2 \\ \vdots \\ X_N \end{pmatrix}$$

where each X_i, $1 \leq i \leq N$ is a time series on its on. The size of these time series can vary.

In the multidimensional case, clustering involves grouping entities of the form X. Figure 2 shows an example of hierarchical clustering for 2-dimensional time series (there are 5 entries or instances that are clustered).

FIGURE 2. Two dimensional time series: example of hierarchical clustering.



Multi-dimensional time series occur if one deals with multiple measurements on some objects, phenomena, or variables.

In time series clustering, each item in the set of items to be clustered is a series of records in time. For instance, the temperatures measured each

day, over the course of 2 years in a certain city are a time series. The same measurements for a number of cities represent the set of the time series which are to be clustered, based on the temperature values recorded in two years.

Each data in this case consists of 730 points (two years of 365 days each) in a two dimensional space. In the multidimensional case, each data consists of more than just one time series. For instance, we want to clusters cities which are not similar only with respect to temperature values over the course of two years, but also the wind speed, pressure, precipitations volume, etc, each of them measured daily. The figures below show some examples of items having two time series each. Some of them may be more similar with respect to one of the time series, while the other will be more similar with respect to the other. In multidimensional clustering we want to cluster together items which are similar with respect to all the time series, regarded in general. This example is illustrated in Figure 3.

Many times, the multidimensional time series data are converted into a single time series by concatenating all the time series into a single one. But this will conduct to loss of generality. The advantage of dealing with a multidimensional time series as such without transforming them is that, on the one hand, it offers a global point of view and shows some critical pathologies arising from evident discrepancies, whereas, on the other hand, it permits to integrate the information contained in each one-dimensional time series of X and therefore it is useful when each array is sparse and short [23].

4. A variant of multidimensional time series data clustering

The similarity between two time series is usually calculated using a distance or a similarity measure. In this section we consider the difference between each time series (of a multidimensional time series instance) as an objective function which has to be minimized. Thus, for comparing how similar two objects X and Y are, where X and Y are given by:

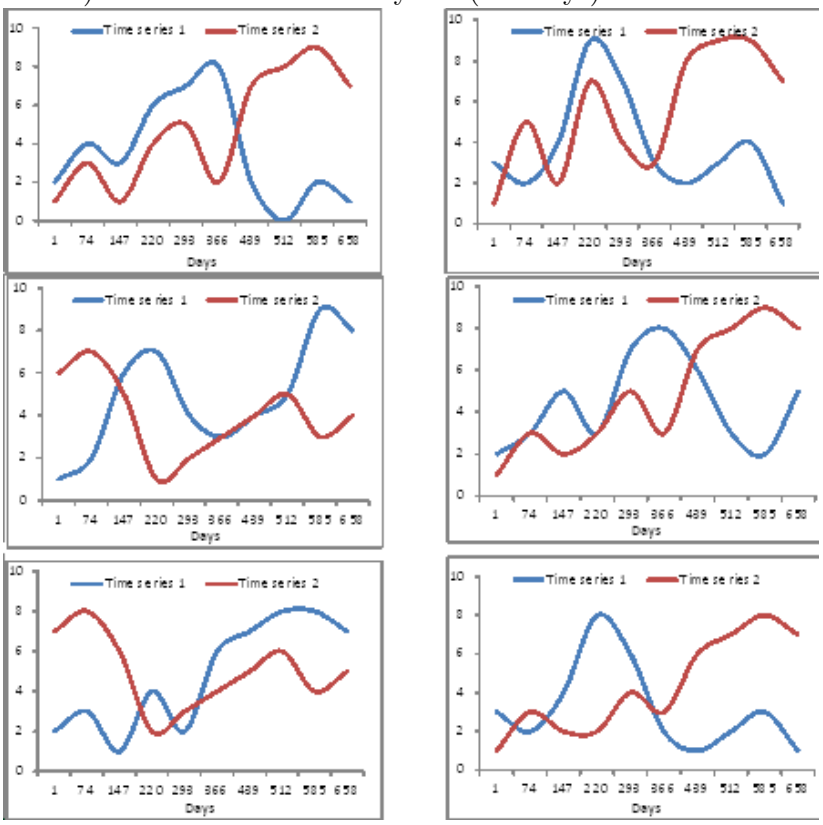
$$\mathrm{X} = \begin{pmatrix} X_1 \\ X_2 \\ \vdots \\ X_N \end{pmatrix}, \mathrm{Y} = \begin{pmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_N \end{pmatrix}$$

we define an N dimensional objective function F= (f_1, f_2, …, f_N) as:

$$\mathrm{F} = \begin{pmatrix} f_1 = d\,(X_1,\, Y_1) \\ f_2 = d\,(X_2,\, Y_2) \\ \vdots \\ f_N = d\,(X_N,\, Y_N) \end{pmatrix}$$

where d(·) defines a similarity measure.

FIGURE 3. Example of two sets of measurements (two time series) over the course of two years (730 days).



We use k-means for clustering multidimensional time series data. In our case, each item is assigned to a cluster based on the values of the F function. We consider a weighted combination of all f_i,1≤ i ≤ N as a result of the similarity and denote this by d_{sim}:

$$d_{sim} = \sum_{i=1}^{N} w_i f_i$$

where w is a vector of weights denoting the importance of that particulate time series in the clustering. For our experiments we considered all time series as having equal importance and in this case w_i =1, 1≤ i ≤ N.

We implemented four different distances d(·):

- Euclidean distance
- Manhattan distance

TABLE 1. Weather records

Algorithm	Number of clusters	Silhouette coefficient
k-means with Euclidean distance	8	0.2105
k-means with Manhattan distance	10	0.1574
k-means with Maximum distance	12	0.0200
k-means with Average distance	8	0.2105

- Maximum distance
- Average distance

Setting the value of k. One of four distance measures (Euclidean distance, Manhattan distance, Maximum distance, Average distance) is selected from the main menu, and sent as parameter for the algorithm to use while computing. Also a Maximum Distance Percent can be introduced before running the algorithm; the default value for this variable is 0.6 in our experiments.

The algorithm starts with a large k (equal to the no. of items to cluster) which is decreased step-by-step (by moving data, if convenient, from initial clusters - containing only one item from the data set - to new clusters - containing similar items) until it reaches a value that satisfies the stability of each cluster (small distance between data belonging to same cluster, large distance between data belonging to distinct clusters).

4.1. **Numerical experiments.** We perform experiments considering three datasets from various domains. Silhouette coefficient [2] is used to compare the performance of k-means for various distance measures.

4.2. **Weather records data set.** This data set contains data about countries with respect to temperature, precipitation level, atmospheric pressure and humidity. The countries have to be clustered based on the records over time for all these parameters together.

The details of the data set are:

- 14 (Countries);
- No. of parameters: 5 (Precipitations Level (L/mˆ2), Wind Speed (m/s), Temperature (grC), Atm. Pressure (mmHg), Humidity (%RH));
- No. of time points: 77.

The results obtained by k-means are presented in Table 1.

From the experiments we observe that:

- Best average silhouette coefficient: Euclidean distance and Average distance;

- Better average silhouette coefficient for cluster 0 is obtained using Manhattan distance (0.745) not Euclidean/Average distance (0.181) or Maximum distance (0.240);
- Better average silhouette coefficient for cluster 1 is obtained using Euclidean distance or Average distance (0.674);
- Best average silhouette coefficient obtained for a cluster is 0.828 using Euclidean, Average or Manhattan distance.

4.3. **Sensors recordes data set.** This data set if from the Machine Learning Repository. The file contains 19 activities (like sitting, lying on back and on right side, ascending and descending stairs, running on a treadmill with a speed of 8 km/h, etc). Data is acquired from one of the sensors (T_xacc) of one of the units (T) over a period of 5 sec, for each subject and for each of the activities.

Results obtained by k-means are presented in Table 2. In this case we tested the algorithm with two values for the maximum Distance Percent parameter (used to decide which k (number of clusters) is best): 0.6 and 0.9.

We observed that:

- Best average silhouette coefficient: Manhattan distance using Max Distance Percent 0.9;
- The same average silhouette coefficient for cluster 1 is obtained using Manhattan distance, Euclidean distance or Average distance and the default Max Distance Percent (0.6) or Average distance and a Max Distance Percent = 0.9 (0.521)
- The same average silhouette coefficient for cluster 0 is obtained using Maximum distance and the default Max Distance Percent or Euclidean distance or Average distance and a Max Distance Percent = 0.9 (0.491);
- Best average silhouette coefficient obtained for a cluster is 0.613 using Manhattan distance and Max Distance Percent 0.9;
- For Max Distance Percent lower than default (0.6) worse clustering results have been obtained.

4.4. **KEGG biological data set.** The third dataset if from the KEGG [24]database and is not a time series dataset. We wanted to test the algorithm for this kind of data as well, in order to validate the findings. The data is a Metabolic Relation Network (Directed) Data Set. It has 8 attributes such as: Nodes (min:2, max:116), Edges (min:1, max:606), Connected Components (min:1, max:13), Network Diameter (min:1, max:30), Network Radius (min:1, max:2), Shortest Path (min:1, max:3277), Characteristic Path Length (min:1), Average number of Neighbors (min:1))

TABLE 2. Sensors recordes

Algorithm	Number of clusters	Silhouette coefficient
Max Distance Percent = 0.6		
k-means with Euclidean distance	18	0.028
k-means with Manhattan distance	18	0.028
k-means with Maximum distance	17	0.028
k-means with Average distance	18	0.028
Max Distance Percent = 0.9		
k-means with Euclidean distance	17	0.028
k-means with Manhattan distance	16	0.038
k-means with Maximum distance	17	0.028
k-means with Average distance	18	0.028

The data set has 1,000 instances.
The results obtained by k-means are given in Table 3.

TABLE 3. KEGG data set

Algorithm	Number of clusters	Silhouette coefficient
k-means with Euclidean distance	618	0.0014
k-means with Manhattan distance	618	0.0014
k-means with Maximum distance	618	0.0014
k-means with Average distance	618	0.0014

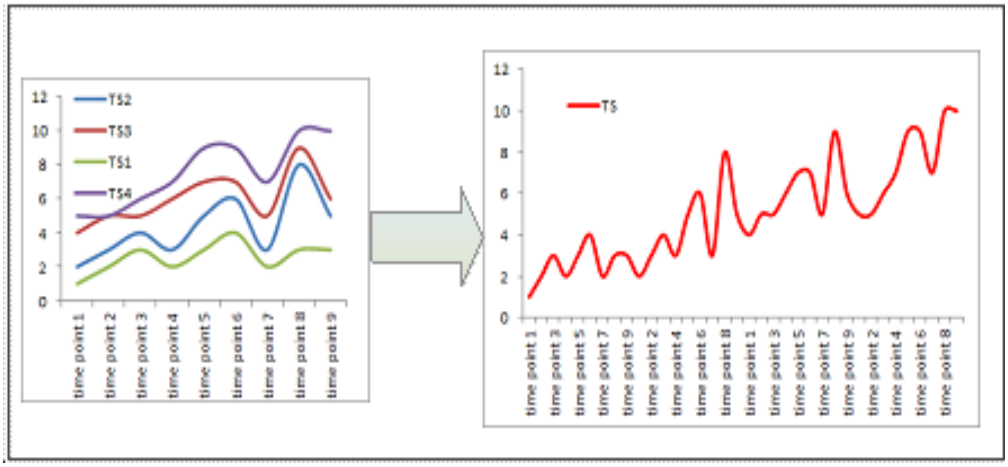
We can observe that:

- The same average silhouette coefficient is obtained for all distance measures (0.0014);

- Using different values for Max Distance Percent (0.2, 0.6, 0.9) hasn't improved the results;
- The best average silhouette coefficient obtained for a cluster is 0.920.

4.5. **Comparison with traditional methods.** In the traditional methods the data is usually pooled, that is a single parameter is inferred for all time series. In this way, a multi-dimensional time series item is transformed into a single dimensional one as it can be seen in Figure 4. We have implemented this approach and tested it using the same settings and under the same conditions as for our approach. For the first data set, for two of the similarity metrics – Manhattan and Average – the number of clusters obtained by the traditional methods was higher that the real one. For the average measure in the second data set the number of clusters obtained by the traditional methods was again higher.

FIGURE 4. Comparison with traditional methods



5. CONCLUSIONS

This paper investigates the role of various distance measures in k-means algorithm for clustering multidimensional time series data. Euclidean distance is the most frequent used and most common measure. Our experiments on – three different data sets – reveal that Manhattan distances (and sometimes the average distance) are better candidates for similarity between two multidimensional time series instances. This work only investigates geometrical

distances, but as future work, geometric distances presented here will be compared with other similarity measures (such as descriptive measures, pattern finding measures, etc.).

References

[1] Evertitt B. *Cluster analysis 3rd edition* New York, Toronto: Halsted Press, 1993

[2] Kaufman L., Rousseeuw P. *Finding Groups in Data: An Introduction to Cluster Analysis* Wiley Series in Probability and Mathematical Statistics. New York: John Wiley and Sons, Inc.,1990.

[3] Williams W., Lammbert J. *Multivariate methods in plant ecology: V. similarity analyses and information-analysis* Journal of Ecology, 54(2):427445, 1966.

[4] Duran B., Odell P. *Cluster Analysis: A Survey*, volume 100 of Lecture Notes in Economics and Mathematical Systems Berlin, Heidelberg, New York: Springer-Verlag,1974.

[5] Lance G., Williams W. *A general theory of classificatory sorting strategies I. Hierarchical systems* The Computer Journal , 9(4):373380.

[6] Florek K., Lukaszewicz J., Steinhaus H., Zubrzycki S. *Sur la liaison et la division des points d'un ensemble fini* Colloquium Mathematicum, 2:282285.

[7] Johnson S. *Hierarchical clustering schemes* Psychometrika, 32(3):241254.

[8] Guojun G., Chaoqun M., Jianhong W. *Data Clustering: Theory, Algorithms and Applications* SIAM Publishers, 2007.

[9] Zhang B., Srihari S. *Properties of Binary Vector Dissimilarity Measures* Technical Report, CEDAR, Department of Computer Science and Engineering, University of Buffalo, The State University of New York, 2003. http://www.cedar.buffalo.edu/papers/publications.html.

[10] Macqueen J. *Some methods for classification and analysis of multivariate observations* Proceedings of the 5th Berkeley symposium on mathematical statistics and probability, volume 1, pages 281–297. Berkeley, CA: University of California Press.

[11] Jain A., Dubes R. *Algorithms for Clustering Data* Englewood Cliffs, NJ: PrenticeHall.

[12] Hartigan J., Wong M. *Algorithm AS136: A k-means clustering algorithm* Applied Statistics, 28(1):100–108.

[13] Hartigan J. *Clustering Algorithms* Toronto: JohnWiley & Sons.

[14] Phillips S. *Acceleration of k-means and related clustering algorithms. In Mount, D. and Stein, C., editors,* ALENEX: International workshop on algorithm engineering and experimentation, LNCS, volume 2409, pages 166–177. San Francicsco: Springer-Verlag.

[15] Kendall S., Ord J. *Time Series, 3rd edition* Seven Oaks, U.K.: Edward Arnold.

[16] Tsay R. *Analysis of Financial Time Series* Wiley Series in Probability and Statistics. New York: JohnWiley & Sons.

[17] Shadbolt J., Taylor J. *Neural Networks and the Financial Markets* London: Springer**20**.

[18] Azoff E. *Neural Network Time Series Forecasting of Financial Markets* New York: JohnWiley & Sons.

[19] Gunopulos D., Das G. *Time series similarity measures (tutorial PM-2)* Tutorial notes of the sixth ACM SIGKDD international conference on knowledge discovery and data mining, pages 243–307. Boston, MA: ACM Press .

[20] Bollobás B., Das G., Gunopulos D., and Mannila H. *Time-series similarity problems and well-separated geometric sets* SCG '97: Proceedings of the thirteenth annual symposium on computational geometry, pages 454–456. New York: ACM Press.

[21] Das G., Gunopulos D., Mannila H. *Finding similar time series* Proceedings of the first European symposium on principles of data mining and knowledge discovery, pages 88–100. New York: Springer-Verlag.
[22] Warren Liao T. *Clustering of time series data—a survey* Pattern Recognition, Volume 38, Issue 11, Pages 1857-1874,2005.
[23] Franciosi M., Menconi M. *Multi-dimensional sparse time series: feature extraction* CoRR abs/0803.0405 (2008).
[24] Kanehisa, M. *The KEGG database. In Novartis Found Symp* 2002, January, Vol. 247, pp. 91-101.

Babeş-Bolyai University, Department of Computer Science, 1, M. Kogălniceanu street, 400084 Cluj-Napoca, Romania
E-mail address: ramona@cs.ubbcluj.ro

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LIX**, Number 1, 2014

POST PROCESSING VOTING TECHNIQUES FOR LOCAL STEREO MATCHING

ALINA MIRON

Abstract. In this paper we propose two extensions to the Disparity Voting scheme for local stereo matching algorithms, that improve the quality of the disparity map. These extensions are based on two separate hypothesis on the disparity map: the real disparity value of a pixel is found close to the disparity value that gives the minimum matching cost and the real disparity value of a pixel is not always the one given by the minimum *cost*, but nevertheless is found among one of the minimum matching *costs*. These techniques are compared on a real road scene dataset (KITTI) against the classical Disparity Voting and Winner-Takes-All strategy.

1. Introduction

Stereo vision refers to the extraction of depth information from a scene when viewed by a two camera system (eg. human eyes). When an object is viewed from a great distance, the optical axes of both eyes are parallel, therefore the object's projections, as seen by each eye independently, is similar. On the other hand, when the object is placed near the eyes, the optical axes will converge. The main application of human stereo vision is the perception of depth, while computer stereo vision has applications that vary from 3D reconstruction to image-based rendering or object hypothesis generation.

A task that is learned so easily by the human brain and performed unconsciously has proven to be difficult for computers. In traditional computer stereo vision, two cameras are placed horizontally at a certain distance in order to obtain different views of the scene. The distance between the cameras is called baseline and influences the minimum and maximum perceived depth.

Received by the editors: April 28, 2014.

2010 *Mathematics Subject Classification.* 68T45.

1998 *CR Categories and Descriptors.* I.2.10 [**ARTIFICIAL INTELLIGENCE**]: Vision and Scene Understanding – *3D/stereo scene analysis*; I.4.8 [**IMAGE PROCESSING AND COMPUTER VISION**]: Scene Analysis – *Stereo*.

Key words and phrases. stereo vision, disparity refinement, road scene analysis.

The amount to which a single pixel is displaced in the two images is called disparity and it is inversely proportional to its depth in the scene: closer objects will have greater disparity than background objects.

The field of application that we target is the one of intelligent vehicles, in particular the detection of road obstacles like pedestrians. A robust and accurate disparity map is essential in order to have pertinent information over the location of pedestrians in a scene and the relative distance from the vehicle to those pedestrians.

As presented by [11], most of the stereo matching algorithms rely on four important steps: Cost computation; Cost aggregation; Disparity computation/optimisation and Disparity refinement. Each step is important for the quality of the disparity map, with the cost computation step being crucial as it stands at the basis of the stereo matching algorithms.

There exists several studies where comparison of cost functions is performed, the most extended ones being made by Hirschmuller and Scharstein [2, 3]. While there are different studies that cover the step of Cost Computation, from our knowledge, the step of Disparity computation for local algorithms has been overlooked. In [7], it has been shown that different refinement techniques can greatly improve the accuracy of the obtained disparity map. Unfortunately, most of the disparity refinement techniques rely on computing two disparity maps, one for left and the other one for right image, leading to an increase in computation time. This is followed by techniques as left-right consistency check in order to refine the disparity results. We argue that the disparity map can be improved starting from the initial disparity estimation.

This article is organized as follows: we start by briefly describing the stereo matching algorithms employed in section 2, while in section 3 are presented the two disparity voting techniques proposed. We continue with a description of the results obtained (section 4) and we conclude with a discussion about the impact of the proposed techniques.

2. Stereo Matching Algorithm

Cost function. For the cost function we are going to use *DiffCensus* proposed in [8]. *DiffCensus* has proven to be robust to radiometric distortions, and in the same time provides better results on road scene images than other cost functions. The main advantage of the function is that it does not rely on the value of the pixel intensity but on the difference of intensity between a considered pixel and its neighbourhood (equation 1). This keeps the function as a non-parametric one while incorporating extra information.

$$C_{DIFFCensus}(x,y,d) = \rho(C_{census}(x,y,d),\lambda_{census}) + \rho(C_{DIFF}(x,y,d),\lambda_{DIFF}) \tag{1}$$

where C_{census} is the Census transform cost function proposed by [13], x, y are the coordinates of the considered pixel, d is the disparity value, λ_{census} and λ_{DIFF} are user defined constants, while ρ is defined in eq 2 and C_{DIFF} is defined in eq. 3 .

$$\rho(c,\lambda) = 1 - exp(-\frac{c}{\lambda}) \tag{2}$$

$$C_{DIFF}(x,y,d) = |\overline{DIFF}(x,y) - \overline{DIFF}(x,y-d)| \tag{3}$$

where $n \times m$ is the same support window that is used to compute the CT.

$$\overline{DIFF}(u,v) = \frac{DIFF(u,v)}{CensusSize} \tag{4}$$

where *CensusSize* is the size of the bit string given by the support window $n \times m$ and *step*.

$$DIFF(u,v) = \sum_{\substack{i=\overline{1:step:n} \\ j=\overline{1:step:m}}} (|I(u,v) - I(u+i,v+j)|), \tag{5}$$

where $I(u,v)$ is the intensity of the pixel located at coordinates (u,v).

Cost aggregation. Zhang et al. [14] proposed an efficient technique based on cross-zone aggregation for computing a pixel aggregation region, that takes into consideration color and euclidean distances.

The idea behind is to construct a *cross* region for each pixel. For this, it is necessary to find only four pixels, corresponding to the end of the four arms: up, down, left and right (figure 1.a). Then, in order to construct a region of various shapes, for each pixel that lies on the vertical arm, the horizontal arm will give the region boundaries for the specific row.

In order to choose an arm endpoint p_e for a given pixel $\mathbf{p}$, two rules are applied that pose limitations on color similarity and maximum arm length:

- $D_c(p_e,p) < \tau$. τ is a user-defined threshold value, while the color difference is defined to be $D_c(p_e,p) = max_{i\in R,G,B}|I_i(p_e) - I_i(p)|$.
- $D_s(p_e,p) < L$. L is a user-defined threshold value and represents a maximum length in pixels. $D_s(p_e,p)$ is a spacial distance given by $|p_e - p|$.

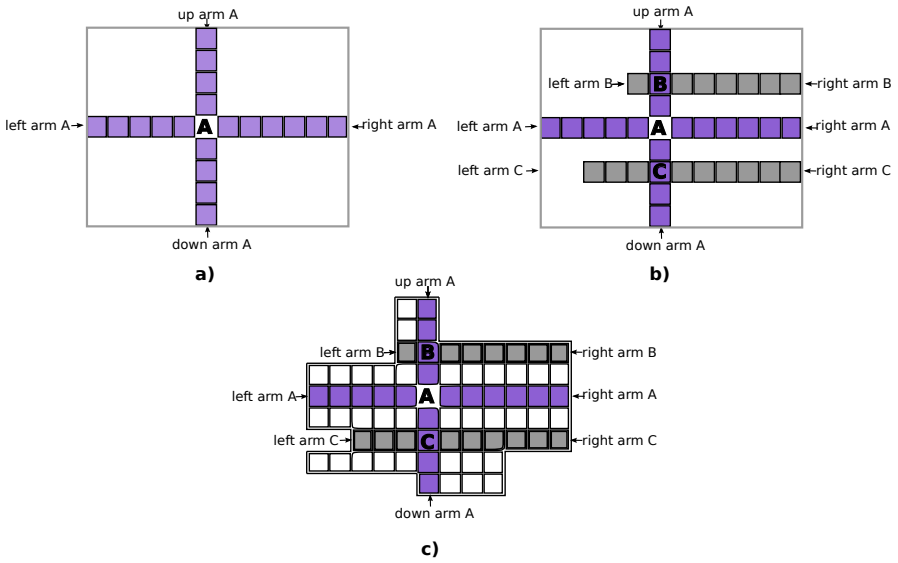


FIGURE 1. Cross region construction: **a)** For each pixel four arms are chosen based on some color and distance restrictions; **b)**,**c)** The cross region of a pixel is constructed by taking for each pixel situated on the vertical arm, its horizontal arm limits.

After having the cross region for each pixel, the next step is to compute the cost in the defined region. For this, the cost aggregation is computed in two steps. First the horizontal matching cost is computed and stored, secondly the final cost is obtained by aggregating the intermediate results vertically. The two steps can be efficiently computed using $1D$ integral images.

As comparison for the local stereo matching algorithm based on cross-zone aggregation, we are also going to use a global stereo matching algorithm based on Graph Cuts. As described by Kolmogorov and Zabih [4], a graph cut is a partition of a graph with two distinguished terminals called source (s) and sink (t) into two sets V^s and V^t, such that $s \in V^s$ and $t \in V^t$. The cost of the cut is represented by the sum of the edges' weights between the two partitions. Finding the minimum cut (the cut of minimum costs among all possible cuts), and implicitly the minimum cost, can be resolved by computing a maximum flow between terminals. In practice, the global energy minimisation technique using graph cuts has been shown to be effective with the condition of having an appropriate cost function. The same *DiffCensus* cost function was used also for the graph cuts algorithm.

Disparity computation. As reference point we are going to consider two strategies for choosing the disparity: a classical one, Winner-Takes-All (WTA) and disparity voting.

The simplest strategy of choosing a disparity for a given pixel is the WTA strategy, i.e. finding the point that will minimize the matching cost (see equation 6). Unfortunately, this does not always give the best results. This is due to the fact that even if a certain disparity has the minimum matching cost, this does not make it necessarily the right disparity (stereo matching is an NP-complete problem).

$$d_p = min_{d_{min} \leq d \leq d_{max}} \sum_{q \in N_p} c(q, q-d) \tag{6}$$

where

- d_p is the final disparity assigned to pixel p
- d_{min} and d_{max} is the minimum possible disparity, respectively maximum.
- N_p represents the neighbourhood of pixel p that is taken as aggregation area
- $c(q, q-d)$ represents a cost between the pixel q in the left image and the corresponding pixel at disparity d in the right image

Contrary to WTA, a simple strategy is the disparity voting strategy proposed in [6] and reused in [7, 14]. The aggregation area will usually originate from the same scene patch. Therefore, the pixels in an aggregation area should share similar disparities. For every pixel p, having a disparity estimate d_p cumputed with WTA, a histogram h_p of disparities is build as showed by equation 7:

$$h_p(d) = \sum_{q \in U(p)} \delta(d_q, d) \tag{7}$$

where $U(p)$ represents the set of all aggregation areas that contain the pixel p, and the function δ is defined as follows:

$$\delta(d_a, d_b) = \begin{cases} 1 & \text{if } d_a = d_b \\ 0 & \text{otherwise} \end{cases}$$

$$d_p^* = argmax(h_p(d)) \tag{8}$$

where $d \in [0, d_{max}]$.

3. Proposed Voting Strategies

We extend the voting strategy proposed by [6] using two different assumptions:

- The real disparity of a pixel is found close to the disparity corresponding to the minimum matching cost (see figure 2.a))
- The real disparity of a pixel is given by one of the disparities that have a matching cost close to the minimum matching costs (see figure 2.b)).

Different from Zhang et al. [14], we propose an extension for the voting algorithm. Due to the fact that different but close disparities have similar matching costs, the surface of inclined objects will not appear very smooth. This corresponds to the first presented assumption. Our proposal is for the voting scheme to not only consider the disparity d_p obtained with WTA, but also the disparities in the interval $[d_p - v, d_p + v]$, as presented in equation 9. We will further refer to the strategy given by this assumption as *VotingInInterval*.

$$h_p(d) = \sum_{d \in [d-v, d+v]} \sum_{q \in U(p)} \delta(d_q, d) \tag{9}$$

In certain situations the minimum cost might not give the real disparity value. Moreover, there exist cases where the same minimum cost is shared by multiple disparity values. Therefore we can use the second assumption in order to model the voting strategy: the voting scheme will not only consider the disparity d_p obtained with WTA, but also the disparities that have close matching cost of d_p, as presented in equation 10. We will further refer to the strategy given by this assumption as *VotingMinCosts*.

$$h_p(d) = \sum_{d \in M_{\overline{1,v}}} \sum_{q \in U(p)} \delta(d_q, d) \tag{10}$$

where M represents a sorted set of disparity using as criterion the matching cost. M_1 corresponds to the disparity that has the minimum matching cost, therefore d_p.

4. Experiments

There exist several challenging databases for testing the stereo matching algorithms, from simulated road scenes like *Van Synthetic stereo* [12] and EISATS [9], to real road scenes with some degree of ground truth like KITTI [1], Make3D Stereo [10] or Ladicky[5]. Moreover one of the most well know benchmark for the stereo matching algorithms is the Middlebury[11] dataset. From all these datasets, the most used datasets are Middlebury, that contains

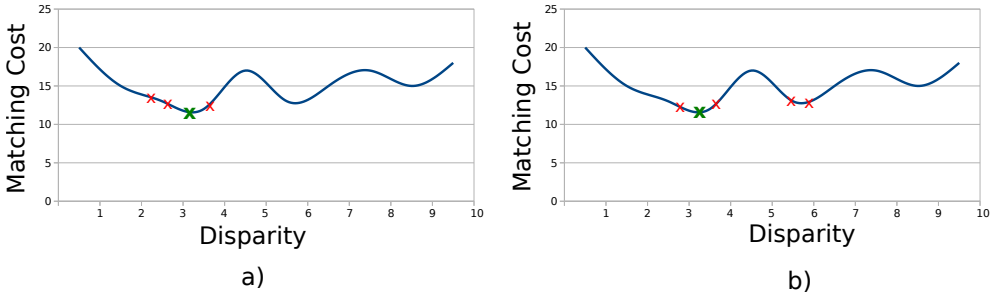


FIGURE 2. Proposed voting strategies: a) the first one uses voting strategy in an interval of disparities around the minimum matching cost (showed in green) b) while the second one uses a voting strategy for the disparities that give matching cost close to the minimum matching cost

images taken indoors in controlled light conditions, and KITTI that has real road scenes images.

In what follows, we use the KITTI stereo images for all the numerical experiments. KITTI dataset is divided into 194 images in the training set for which the ground truth images is provided, and 195 images in the testing set for which an evaluation server should be used in order to obtain the results. The following experiments are performed only on the 194 images in the training set[1] All the cost functions in this paper are evaluated by the average percentage of erroneous pixels in all zones, occlusions included, and computed at 3 pixels error threshold.

In table 1 is presented a comparison of obtained error rates on the KITTI dataset using cross-zone aggregation, the cost C_{DIFFCF}, and five strategies for deciding the final disparity: WTA, the voting method proposed by Zhang et al. [14], a global method Graph Cuts [4] and the two proposed voting strategies. For the proposed voting strategies the parameter v was empirically chosen to be *two* for VotingInInterval strategy and *six* for VotingMinCosts strategy. It can be observed that by simply adding the votes to a disparity interval rather than just one disparity values the error rate decreases with 2.2%. Adding the votes not just for the minimum matching cost, but rather for all the disparity values that give a matching cost close to the minimum one, also improves the results, but not as much as the voting interval strategy.

[1]At the moment of performing the tests, only one submission in 72 hours was allowed on the evaluation server.

Disparity Decision Strategy	**Error Rate**
Winner Takes-All	15.05%
Voting Zhang et al. [14]	12.70%
Graph Cuts	13.05%
Proposed VotingInInterval (v=2)	**10.50**%
Proposed VotingMinCosts (v=6)	11.05%

TABLE 1. Comparison of different strategy methods for choosing the disparity

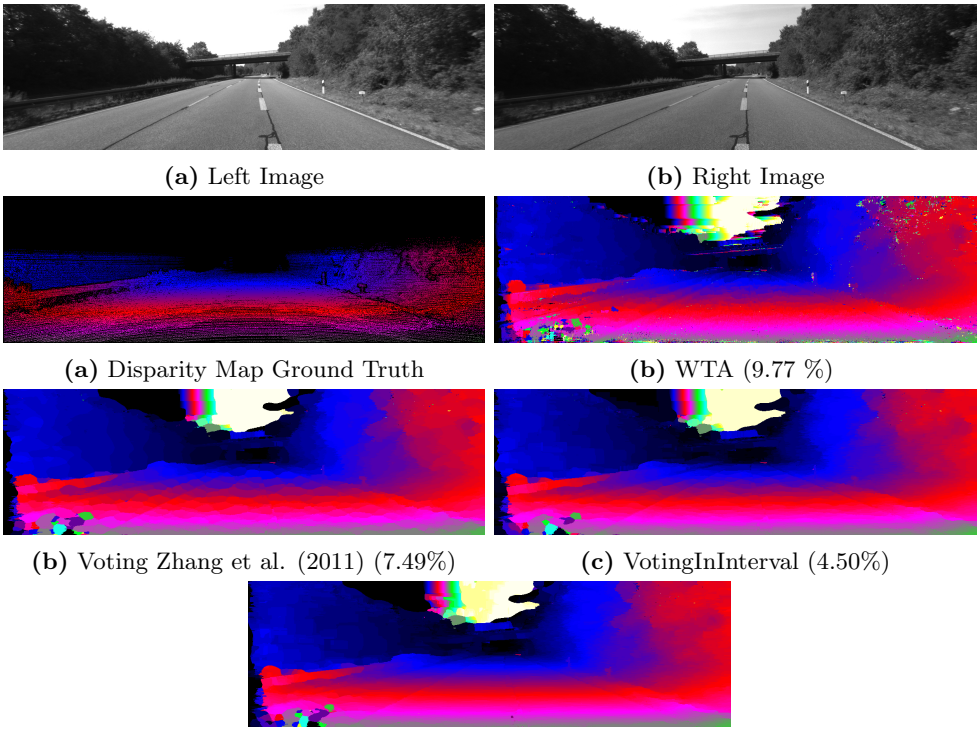
In figure 3 is presented a visualisation of the disparity map produced by the compared algorithms: WTA, Voting Zhang et al. (2011) , and the two proposed Voting strategies (VotingInInterval and VotingMinCosts). From the presented images, it can be observed that the VotingMinCosts produces the smoothness disparity map, but the overall better results are obtained by the VotingInInterval strategy.

5. Conclusions

In this paper we have presented two new Voting strategies for the step of Disparity computation for local Stereo Matching algorithms. These are based on two rather simple assumptions about the disparity map. The proposed strategies have proven that they are capable to improve the disparity map results. In comparison with other techniques of disparity refinement, because the decision is taken at the moment of disparity computation, both strategies come with a very low computation cost. As future work, the two strategies could be combined in order to take advantage of both voting mechanisms. Also, these strategies can be used in combination with any stereo matching algorithms. Thus, it would be interesting a comparison across multiple stereo vision algorithms in order to see if the performance gain remains always the same.

References

[1] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Computer Vision and Pattern Recognition (CVPR)*, Providence, USA, June 2012.



(a) Left Image **(b)** Right Image

(a) Disparity Map Ground Truth **(b)** WTA (9.77 %)

(b) Voting Zhang et al. (2011) (7.49%) **(c)** VotingInInterval (4.50%)

(d) VotingMinCosts (6.9%)

FIGURE 3. Different Disparity maps produced by different Voting Strategies for the same image

[2] Heiko Hirschmuller and Daniel Scharstein. Evaluation of cost functions for stereo matching. In *IEEE Computer Vision and Pattern Recognition (CVPR)*, pages 1–8, 2007.

[3] Heiko Hirschmuller and Daniel Scharstein. Evaluation of stereo matching costs on images with radiometric differences. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(9):1582–1599, 2009.

[4] Vladimir Kolmogorov and Ramin Zabih. Computing visual correspondence with occlusions using graph cuts. In *IEEE International Conference on Computer Vision*, volume 2, pages 508–515, 2001.

[5] Lubor Ladickỳ, Paul Sturgess, Chris Russell, Sunando Sengupta, Yalin Bastanlar, William Clocksin, and Philip HS Torr. Joint optimization for object class segmentation and dense stereo reconstruction. *International Journal of Computer Vision*, pages 1–12, 2012.

[6] Jiangbo Lu, Gauthier Lafruit, and Francky Catthoor. Anisotropic local high-confidence voting for accurate stereo correspondence. In *Proc. SPIE-IS&T Electronic Imaging*, volume 6812, pages 605822–1, 2008.

[7] Xing Mei, Xun Sun, Mingcai Zhou, Shaohui Jiao, Haitao Wang, and Xiaopeng Zhang. On building an accurate stereo matching system on graphics hardware. In *IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, pages 467–474, 2011.

[8] Alina Miron, Samia Ainouz, Alexandrina Rogozan, and Abdelaziz Bensrhair. A robust cost function for stereo matching of road scenes. *Pattern Recognition Letters*, 38:70–77, 2014.

[9] Sandino Morales and Reinhard Klette. Ground truth evaluation of stereo algorithms for real world applications. In *Computer Vision–ACCV 2010 Workshops*, pages 152–162. Springer, 2011.

[10] Ashutosh Saxena, Jamie Schulte, and Andrew Y Ng. Depth estimation using monocular and stereo cues. IJCAI, 2007.

[11] Daniel Scharstein and Richard Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International journal of computer vision*, 47(1):7–42, 2002.

[12] Wannes Van Der Mark and Dariu M Gavrila. Real-time dense stereo for intelligent vehicles. *IEEE Transactions on Intelligent Transportation Systems*, 7(1):38–50, 2006.

[13] Ramin Zabih and John Woodfill. Non-parametric local transforms for computing visual correspondence. *ECCV*, pages 151–158, 1994.

[14] Ke Zhang, Jiangbo Lu, and Gauthier Lafruit. Cross-based local stereo matching using orthogonal integral images. *IEEE Transactions on Circuits and Systems for Video Technology*, 19(7):1073–1079, 2009.

Insa Rouen, Avenue de l'Université, 76800 Saint-Étienne-du-Rouvray, France; Universitatea Babeş-Bolyai, Facultatea de Matematică şi Informatică, Str. Mihail Kogălniceanu, nr. 1, 400084 Cluj-Napoca

E-mail address: alina.miron@insa-rouen.fr

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LIX**, Number 1, 2014

ON UML BASED NOTATIONS FOR ASPECTS

GRIGORETA S. COJOCAR AND ADRIANA M. GURAN

Abstract. Separation of concerns was always an important factor in designing easily maintainable and evolvable software systems. However, practice has shown that it is not easy to clearly separate all the concerns from a software system in analysis, design and implementation phases. Using just one programming paradigm most concerns can be clearly separated, but there are still a few concerns whose design and implementation is entangled with other concerns. New programming paradigms that try to complement the existing ones by providing new ways for a better separation of concerns have been developed. Aspect oriented paradigm is one of the new paradigms. Although the number of languages supporting aspect oriented programming has increased, there is no generally accepted notation for aspects in UML. In this paper we provide an analysis of the existing UML-based notation for aspects in the class diagram using a set of comparison criteria.

1. Introduction

Separation of concerns [20] was always an important factor in designing easily maintainable and evolvable software systems. However, practice has shown that it is not easy to clearly separate all the concerns from a software system. Using just one programming paradigm most concerns can be clearly separated, but there are still a few concerns whose design and implementation is entangled with other concerns. New programming paradigms that extend the existing ones in order to better separate the concerns that are still entangled have been developed. Among these we mention the aspect oriented paradigm (AOP) that usually extends the object-oriented paradigm [18].

There are already a number of aspect oriented languages as AspectJ [4] and AspectC++ [3] that provide new language constructs for implementing the crosscutting concerns. At this moment there is no generally accepted design

Received by the editors: May 5, 2014.

2010 *Mathematics Subject Classification.* 68N19, 68N99.

1998 *CR Categories and Descriptors.* D.2.2[**Software Engineering**]: Design Tools and Techniques – *Computer-aided software engineering*; D.1.m [**Software**]: Programming Techniques – *Miscellaneous*.

Key words and phrases. aspect oriented paradigm, design, UML class diagram.

notation that supports the design of aspect oriented systems. The need for a design notation is justified by the following advantages:

- it would ease the development of aspect oriented systems;
- it would make the understanding of an aspect oriented system much easier (by using a graphical representation);
- it would serve as a basis for assessing the impact of crosscutting concerns on their base classes (core classes);
- it would allow the adaptation and reuse of existing design constructs.

The Unified Modeling Language (UML) is a graphical language for visualizing, constructing, specifying and documenting the artefacts of a software system. The goal of UML is to provide tools for analysis, design, and implementation of software based systems to all parties involved: developers, system architects, etc [7, 22, 28]. One of the main advantages of using UML is that it has defined a set of modeling concepts that are now generally accepted, and it also contains visual representation of the defined concepts that are easy to understand and interpret by humans.

The concepts defined by UML can be used to represent the static structure (such as classes, components, node artifacts) or the behavior (such as activities, interactions, state machines) of the software system [27]. The concepts can be used to build different kind of diagrams (i.e., class diagrams for the static structure and sequence diagrams for the behavior).

Even though UML contains a very large set of concepts, it does not include all the concepts that may appear during the development of different kinds of software systems, mainly because some of the concepts are specific to a certain application domain. That is why, the language also contains extension mechanisms that allow us to add new modeling elements, modify the specification of the existing ones or change their semantics [7].

The extension mechanisms provided by UML are stereotypes, tagged values, constraints, and profiles [27]:

- A *stereotype* extends the vocabulary of the UML, by allowing the creation of new kinds of modeling elements that are derived from existing ones, but that are specific to a particular domain or problem. The information content of the stereotyped element is the same as the existing model element [7, 22].
- A *tagged value* extends the properties of a UML stereotype by allowing the creation of new information in the stereotype's specification [7]. It is a name-value pair that denotes a characteristic of the corresponding stereotype.
- A *constraint* extends the semantics of a UML modeling element allowing addition of new rules or changing the existing ones [7]. The

constraints may be written as free-form text or using the Object Constraint Language (OCL) [29] for a more precise specification of the semantics.

- A *profile* is a UML model with a set of predefined stereotypes, tagged-values, constraints and base classes. It also selects a subset containing only those modeling concepts that are needed and should be used for a particular application area. This mechanism is not considered a first-class extension mechanism, as it does not allow the addition or modification of existing elements. The intention of profiles is to give a simple and easy to use mechanism for adapting an existing set of modeling concepts with constructs that are specific to a particular domain, platform, or method [7, 22].

In UML a *classifier* is a mechanism that describes structural and behavioral features. Classifiers include classes, associations, interfaces, datatypes, components, nodes, use cases, etc. The classifiers that are usually represented in a class diagram are classes and interfaces.

The main contribution of this paper consists in using a new set of comparison criteria for analyzing UML based notations for the aspects displayed in an UML class diagram. The considered criteria are useful in deciding what notation to use for representing aspects as they show the expressiveness of the notation, how easy will be to learn and adopt the new notation.

The paper is structured as follows. In Section 2 we present the new concepts introduced by the aspect oriented paradigm. The considered notations are briefly described in Section 3. An analysis of the described proposals is given in Section 4. Section 5 presents related work. Some conclusions and future research directions are given in Section 6.

2. AOP Concepts

The aspect oriented paradigm introduces new concepts: *join point*, *pointcut*, *advice*, *aspect* and *introduction* for the design and implementation of crosscutting concerns, and *weaving* for building the final software system. Aspect oriented programming can be used only for crosscutting concerns, the core concerns are still designed and implemented using the base programming paradigm, that usually is object-oriented programming, but it can be any other programming paradigm.

2.1. **Join point.** A *join point* is a well-defined point in the execution of a program. Any software systems can be seen as a sequence of execution points like: assignments, conditional statements (if, switch), loop statements (for, while, do-while or repeat), function/method calls, function/methods executions, etc. regardless of the programming paradigm used for developing the

system. Aspect oriented programming only uses some of these points, called join points, in order to add new behavior.

The types of execution points that can be used for developing crosscutting concerns depend on the aspect oriented language. AspectJ [4] offers the greatest variety of execution points, like: method calls, method executions, object instantiations, constructor executions, field references and handler executions, etc., while others allow only a small subset. In Spring AOP [23] a join point always represents a method execution.

2.2. **Pointcut.** The execution of a software system consists of many join points. However, not all of them are necessary for the design and implementation of crosscutting concerns. A *pointcut* selects join points, and exposes some of the values in the execution context of those join points. AspectJ [4] and AspectC++ [3] allow pointcuts to be defined as abstract in order to be able to define reusable aspects and aspect libraries.

2.3. **Advice.** A pointcut allows selecting join points from the software system, however they do not change its behavior. An *advice* defines crosscutting behavior and it is defined in terms of pointcuts. The code of an advice runs at every join point selected by its pointcut. There are different options as to when the code of the advice is executed relatively to the corresponding join point(s). The aspect oriented languages developed so far allow three types: before, after and around the join point.

- *Before*: the advice code is executed before the selected join point. This type of advice does not have the ability to prevent execution flow proceeding to the join point (unless it throws an exception).
- *After*: the advice code is executed after the selected join point. There can be three situations, depending on the execution of the join point:
  - *After returning*: the advice code is executed only if the join point execution completes normally.
  - *After throwing*: the advice code is executed only if the join point execution ends by throwing an exception.
  - *After (finally)*: the advice code is executed regardless of the means by which the selected join point exits (normal or exceptional return). This type of advice is usually called *after finally*, because of its similarity with the `finally` clause of the `try-catch` block from programming languages.
- *Around*: the advice code surrounds the selected join point. It can perform custom behavior before and after the selected join point. It can also decide whether the selected join point should still be executed

or not (called *proceeding*), or it may cause multiple executions of the selected join point.

2.4. **Introduction.** It is sometimes necessary to modify the static structure of a type (by adding new members - attributes/methods or by modifying its inheritance hierarchy) in order to design and implement a crosscutting concern. Even though advices add new behavior to existing types, they do not modify their static structure. An *introduction* allows developers to extend the static structure of existing types. New methods and/or attributes can be added, or the type inheritance hierarchy can be modified (by adding new interfaces or by modifying the base type of the existing type).

2.5. **Aspect.** An *aspect* is a new kind of type specified by the aspect oriented paradigm that is used to implement one crosscutting concern in a modular way. An aspect is similar to a class, it can contain attributes and methods declarations but it also encapsulates pointcuts, advice and introductions. Because of their similarity with classes, an aspect declaration is often almost the same as a new class declaration, where the `class` keyword is replaced with the `aspect` keyword. In some aspect oriented languages aspects (i.e., AspectJ, AspectC++) can inherit from other classes, implement some interfaces or even inherit from other aspects. However some constraints must be followed when inheriting from another aspect. For example, in AspectJ an aspect can inherit only from an abstract aspect.

2.6. **Weaving.** When the aspect oriented paradigm is used for developing software systems, the core concerns are developed independently of the crosscutting concerns. However, in the end, they still have to be put together in order to obtain the final executing system. *Weaving* is the process that produces the final systems, and the *weaver* is the tool used to produce it.

The weaver takes some representation of the core concerns (source code or binaries), some representation of the crosscutting concerns (source code or binaries) and produces the output, which is often a binary representation.

Some aspect oriented languages allow the weaving process to take place at different times. For example, AspectJ allows three different times: compile-time, post-compile time and load-time. Other languages allow only one possibility which is either compile-time or run-time.

The approach used for weaving depends on the aspect oriented language: AspectJ uses byte-code modification, Spring AOP uses dynamic proxies, while AspectC++ uses source code preprocessing.

Not all the concepts introduced by the aspect oriented paradigm can and should be represented in a class diagram. The join points from a software system do not provide any relevant information about the static structure of

the system. However, the visual representation of other concepts can provide useful informtion to the developers. For example, some consider that one of the main difficulties when using the aspect oriented paradigm is that the control flow of the system will be difficult to follow and understand since not all the relevant data about a piece of code can be seen at that code. Some additional information may exist in the aspect that affect that part of code. That is why we consider that adding aspects to the class diagram may ease the understanding of the overall static structure of a software system. The information that should be represented in a class diagram is:

- The aspects that are used for building the system, and their type (abstract or concrete). The internal static structure of an aspect is important as it will show, besides the normal fields and methods, the defined pointcuts together with the collected context, the type of the pointcut (abstract or concret), and the defined advice and their type (before, after, around).
- If and how they change the static structure of other existing elements from the class diagram (classes, interfaces). It should represent the type whose static structure will be modified either by introducing new members (fields, methods, constructors, etc.) or by modifying the inheritance hierarchy of the type (adding a base class or implementing interfaces).
- Relationships with other elements from the class diagram. We consider important the following relationships:
  - Aspect-aspect: An aspect may inherit from another aspect, or an aspect may have precedence over another aspect during weaving.
  - Aspect-interface: An aspect may implement one or more interfaces.
  - Aspect-class: An aspect may inherit (or extend) from a class, it may modify the static structure of an existing class, or it may modify the behavior of a class through one or more advices.

3. Existing UML-based Notations for Aspects

During the last fifteen years, many attempts to identify an appropriate notation of aspect oriented design have been made. Most approaches focused on introducing new modeling elements for the concepts defined by AOP: aspect, advice, pointcut and the relevant relationships, while other approaches (like the one proposed by Herrero et al. [14]) focused on a particular crosscutting concern and tried to introduce special notations for the elements needed to design that crosscutting concern. In this study we have included only the approaches that tried to represented the new concepts introduced by AOP.

Suzuki and Yamamoto were the first to extend the UML with concepts for the design of aspect-oriented programs [25]. The aspect is a stereotype derived from the Classifier element. Aspect can have attributes, operations and relationships. The operations corresponding to an introduction or an advice are represented using the *weave* stereotype followed by the *advice* or *introduce* constraint. The signature of a weave operation also shows which elements (e.g. classes, methods and variables) are affected by the aspect. Relationships of an aspect include generalization, association and dependency. The relationship between an aspect and the classes that the aspect affects is a stereotype of abstraction dependency defined in the UML, called *realize*. They also introduced the stereotype *woven class* into the Class element in order to represent a woven class that should specify the source class and the aspect used to generate it using a tagged value. They do not propose new notations for concepts like pointcut or advice. The proposed model was used to design the Observer design pattern.

Aldawud et al. [2] have proposed an UML profile for aspect oriented modeling. The profile contains an *aspect* stereotype of the Class classifier for representing an aspect, and a *control* stereotype of the Association element for representing relationships, however what kind of relations can be represented using this stereotype is left as a further research question. No notations are provided for join points, pointcuts or advices. They also proposed the use of complex statecharts for modeling aspects behavior [1]. No example of using the profile for designing a crosscutting concern is given.

Kande et al. [17, 16] have first studied the suitability of using UML for representing aspect oriented designs of software system, and then proposed new notations for aspect oriented concepts. The aspect is a stereotype of the Classifier, and a new stereotype of UML collaboration and connection points have been proposed in order to represent the relationships between aspects and other classifiers and pointcuts. Advices and introductions are each represented in their special compartment of the aspect stereotype, and pointcuts are represented as connection points. The proposed design model was used to represent the *Logging* crosscutting concern.

Zakaria et al. [30] have proposed another UML extension for designing aspect oriented systems. The proposal is a more detailed version of the one made by Aldawud et al. in [2]. An aspect is represented as a stereotype of the Class classifier, a pointcut is also modeled as a stereotype of the Class, and advices are represented as stereotypes of the Operation element. They define different stereotypes for each type of advice: before, after, after returning, after throwing and around. For aspect-class relationships they proposed the use of the UML association model element tagged with different values for

different kind of associations (control, track, validate, etc), and for aspect-aspect relationships they either used the UML generalization relationship or the *dominates* stereotype of the Association element. The proposed extension was used to model MoveTracking crosscutting concern.

Pawlak et al. [21] introduced the notion of groups in order to represent objects that are not necessarily having the same class or superclass, but that could be the result of a pointcut selection criteria. An aspect is represented as an aspect-class stereotyped with *aspect*, and it may contain special methods called aspect-methods. These aspect-methods are stereotyped with *before*, *after*, *around*, corresponding to the advice type. To represent pointcuts, they use pointcut relations from aspect-class to another class or group. The pointcut relations are oriented associations stereotyped with *pointcut* and the roles have special semantics. The notation was used to design Authentication and Session aspects.

Stein et al. [24] have proposed an aspect-oriented design model that extends UML using the standard extension mechanism, called AODM. They started from the concepts introduced by AspectJ [4] and they tried to propose the most suited modeling element for each concept. An aspect is a stereotype of the Class classifier (*aspect*), pointcuts and advices are represented as operations of special stereotypes (*pointcut* or *advice*), join points are represented as links (some of the links may be stereotyped, depending on the join point kind), and introductions are represented as special stereotype of collaboration template. They also tried to model the weaving mechanism using a use case stereotype. For aspect-class or aspect-aspect relations they mostly used the already defined relations for the Class classifier with some additional constraints. They also introduced a new relation, called *crosscut*, in order to describe the other modeling elements (classes, interfaces or aspects) that are affected by the associated aspect. The proposed design model was used to design the Observer design pattern [11].

Jacobson and Ng [15] in their book *Aspect-Oriented Software Development with Use Cases* also proposed the extension of UML in order to graphically represent aspects. An aspect is a stereotyped classifier, named *aspect*, with two compartments: one for pointcuts and one for class extensions. Aspects have a class extensions compartment to overlay class extensions onto existing class. With AOP the overlay can be achieved with introductions or advices. Each class extension contains a subset of features (attributes, operations and relationships) that are used to represent which class and which operation is extended by the aspect (using introduction or advice). The pointcuts compartment contains the pointcuts defined by the aspect. The authors also introduced the notion of parametrized pointcuts (marked with $<$ and $>$) that can be used to display an operation extension at multiple join points.

Basch and Sanchez [6] have taken another approach in representing aspects in class diagrams. They have considered the *package* UML element as the most appropriate to be used for aspects. These packages are stereotyped with the *aspect* stereotype. Join points are represented as circles with a cross inside. Inside the aspect package, a class diagram is used to show which other component classes are crosscut by the corresponding aspect. The aspect package should also include interaction diagrams to display the behavior of the aspect. No example of using the proposed approach to design a crosscutting concern is given.

Zhang has also considered using the *package* UML element to represent aspects [31]. The aspect contains two compartments: one for pointcuts and one for advices. The pointcut and the advice are also represented as stereotyped packages. The proposed model was used to design the Logging crosscutting concern.

Some researchers, like Han et al. [13] or Chavez and Lucena [8] have taken the approach of first defining meta-model for AspectJ, and then, based on this metamodel, to define graphical notations for the concepts introduced by AOP.

4. Analysis of UML Based Notations for Aspects

In this section we compare the previously described UML-based notations for aspects from different points of view: the UML representation used for displaying the new concepts introduced by AOP in the class diagram (aspect, advice, pointcut, join point, introduction), the concepts that were represented, the kind of relationships considered between elements from the class diagram (as discussed in Section 2) and statically vizualizing the affected parts of the system.

4.1. **AOP Concepts Represented.**

All the proposals described in Section 3 have considered a way of representing the aspect concept into the class diagram. Most of them also considered representing the advice [17, 21, 24, 25, 30, 31] and the pointcut [15, 17, 21, 24, 31]. Very few proposals considered representing introductions [15, 24, 25] and even fewer considered representing join points [6, 24].

4.2. **UML Notations for AOP Concepts.**

Most of the approaches described in Section 3 represent the aspect starting from the *Class* classifier enhanced with the *aspect* stereotype [2, 21, 24, 30]. Others have considered using the *Classifier* as a base for aspect [15, 17, 25]. Only a few proposals considered using a non classifier as a starting element, namely the package with two compartments for pointcuts and advices [6, 31].

For the rest of the concepts introduced by AOP, there is very little consensus related to the appropriateness of a chosen representation. In the following, we will present the proposed representation for the other concepts:

- *Join point* - There are only two proposals for representing them, consisting in links [24] and a notation in the form of a circle with a cross inside [6].
- *Pointcut* - There is very little overlapping between the proposals for representing pointcuts. In [17] and [24] pointcuts are represented using the *pointcut* stereotype, while Pawlak et al. [21] propose a representation based on an association from an aspect class towards a classifier stereotype with *pointcut*. Zakaria et al. [30] model a pointcut based on the *Class* classifier and by providing a link to its aspect by a *has pointcut* association. Zhang sees it as a stereotype package [31]. Jacobson and Ng [15] proposed representing pointcuts as operations in their own compartment of the aspect stereotype.
- *Advice* - Stein et al.[24] and Zakaria et al.[30] model an advice as an UML operation of a stereotyped named *advice*; while Suzuki and Yamamoto [25] provide a suggestion of using a constraint for the corresponding weave and Kande et al. [17] suggest to represent it as a compartment in the new *aspect* classifier.
- *Introductions* - There is only one approach, proposed by Stein et al. [24], in explicitly modeling introductions that uses parameterized templates. The class extensions compartment introduced by Jacobson et Ng [15] in their proposal can also be used to represent introductions, however from their representation it is not very clear how the extensions will be realized (introduction or advice).

4.3. **Relationships.** Only a few proposals have taken into consideration the relationships that the aspects can have with the other elements from the class diagram (classes, interfaces). Aldawud et al. [2] proposed to use the *control* relationship to represent which other classes the aspect code controls. Suzuki and Yamamoto [25] proposed the usage of the already existing *realize* relationship to represent an aspect and the classes that the aspect affects. Zakaria et al. [30] proposed to use one of the newly introduced *control, track, report, customize, validate, save, handleerror, handleexception* relationships to represent the relation between an aspect and a class. Each aspect should have at least one association with a class in order to affect the system. They also introduced the *dominates* relationship between two aspects in order to represent the precedence between those aspects. Kande et al. [17] introduced the *binding* relationship to specify what class of objects an instance of the aspect can be bound to. Stein et al. [24] introduced the *crosscut* relationship between

an aspect and a class to specify that the aspect affects the class. The crosscut relationship also implies that the aspect requires the presence of the class in order to behave as expected.

4.4. **Vizualising Affected Parts.** Very few proposals considered explicilty vizualizing the parts that will be affected by introducing one or more aspects, however for some proposals the structure of the aspect notation or the relationships introduced can be used for determining the parts of the software system that are affected by the aspects presence. Jacobson and Ng proposal [15] display the affected parts in the *Class extensions* compartment that contains all the classes from the system that will be affected by the aspect (either statically by introductions or dynamically through advice). The aspect notation proposed by Kande et al. [17] contains compartments that display introductions, meaning that those classes will be affected by the aspect. Also, the binding relationship introduced by them show other affected parts (through dynamic crosscutting). The aspect-class relationships described in Section 4.3 can also be considered as relationships that show the affected parts of the software system.

The analysis shows that most approaches have considered representing only a subset of the new concepts introduced by AOP. However in order to properly communicate and describe an aspect oriented design all the concepts should be possible to represent. The results of the analysis also show there is still a need for further investigation in order to represent and include aspect oriented concepts into the UML class diagram of a software system that can be easily adopted by software developers.

5. Related Work

Chitchyan et al. [9] analyzed four proposed approaches that provided support for crosscutting concern at design level: Composition Patterns [10], Aspect-Oriented Component Engineering [12], Hyperspaces [26] and Suzuki and Yamamoto's approach [25]. They defined a set of "good design" criteria that is used for their analysis. The set contains traceability, change propagation, reusability, comprehensibility, flexibility, ease of learning and use, and parallel development.

Asteasuain et al. [5] analyzed the UML suitability for specifying and visualizing aspect oriented design. They considered two types of UML extensions that can be used: a general one that extends the UML with specific concepts from AOP, and a specific one that extends UML only with a few concepts from AOP that are used for the design and implementation of a particular crosscutting concern. From this perspective they compared the extension proposed by Suzuki and Yamamoto [25] considered as a general one, with the extension

proposed by Herrero et al. [14] for the synchronization crosscutting concern, considered as a specific one. The criteria used for comparison are the same with the ones introduced and used by Chitchyan et al. [9] in their analysis.

Lovavio et al. [19] described in their article different notations that were proposed for Aspect Oriented Software Development (AOSD). They first identified the notions used for AOSD (i.e., crosscutting concern, aspect, advice, join point, weaving, relationships, etc.) and then they presented the UML-based notations proposed until that time. However, their focus was not on analyzing and comparing these notations but on using UML for including and modeling aspects early in the software system development life-cycle (not only in the implementation phase).

Our study is different from the above mentioned studies in that it focuses on comparing the way the static structure of an aspect oriented software system can be modeled using UML. Our approach focuses only on the inclusion of aspects in the class diagram, on what AOP concepts can be represented in this diagram and how, and on their relationships with other elements that are displayed in the diagram (classes and interfaces). The first two studies consider other comparison criteria like the ease of tracing crosscutting requirements to design and even to code or how easily the design can be reused for other software systems. The study made by Lovavio et al. also discusses the way the AOP concepts can be represented in UML, but their focus is not mainly on the class diagram. They are interested more in the way UML can be used for identifying and managing crosscutting concerns during requirements and analysis phases.

6. Conclusions and Further Work

We have described in this paper nine existing proposals to introduce aspects into the UML class diagrams of a software system. We have also provided an analysis of the described proposals using different criteria: the aspect oriented concepts represented, the UML notation proposed, the relationships introduced, and if they allow vizualing the affected parts of the software system. Even though there already exist many proposals, none of them is generally accepted nor used in the literature to describe the static structure of an aspect oriented software system. The reasons for this lack of acceptance may be that some of them are incomplete, some of them have as starting point the AspectJ language meaning that some of the features introduced are particular to AspectJ but not to all aspect oriented languages, or the notations proposed are difficult to use for different crosscutting concerns. As described in Section 3, the examples used for presenting the notation are usually small (the Observer pattern, Logging, MoveTracking, Authentication and Session).

Further work should be done in the following directions:

- To propose another notation for representing aspects and their relationships in UML class diagram, starting from the good points of the existing notations.
- To apply the new notation for designing different crosscutting concerns.
- To facilitate the use of the proposed notation by developing a profile or an add-in/plug-in package for an existing CASE tool.

References

[1] Omar Aldawud, Atef Bader, C. Constantinos, and Tzilla Elrad. Modeling intra object aspectual behavior. In *Automating Object-Oriented Software Development Methods Workshop at ECOOP 2001*, pages 1–6, 2001.

[2] Omar Aldawud, Tzilla Elrad, and Atef Bader. A UML Profile for Aspect Oriented Modeling. In *Aspect Oriented Programming Workshop at OOPSLA 2001*, pages 1–6, 2001.

[3] AspectC++ Homepage. http://www.aspectc.org/.

[4] AspectJ Project. http://eclipse.org/aspectj/.

[5] Fernando Asteasuain, Bernardo Contreras, Elsa Estevez, and Pablo R. Fillottrani. Evaluation of uml extensions for aspect oriented design. In *Proceedings of the 4th Ibero-American Symposium on Software Engineering and Knowledge Engineering (JIISIC 2004)*. Madrid, Spain, 2004.

[6] Mark Basch and Arturo Sanchez. Incorporating aspects into the uml. In *Proceedings of the Aspect Oriented Modeling Workshop at AOSD*, 2003.

[7] Grady Booch, James Rumbaugh, and Ivar Jacobson. *The Unified Modeling Language User Guide (2nd Edition)*. Addison-Wesley Professional, 2005.

[8] Christina Chaves and Carlos Lucena. A metamodel for aspect oriented modeling. In *Workshop on Aspect-Oriented Modeling with UML (held in conjunction with the 1st Aspect Oriented Software Development Conference AOSD*, 2002.

[9] Ruzanna Chitchyan, Ian Sommerville, and Awais Rashid. An analysis of design approaches for crosscutting concerns. In *Workshop on Aspect-Oriented Design (held in conjunction with the 1st Aspect Oriented Software Development Conference AOSD*, 2002.

[10] Siobhán Clarke and Robert J. Walker. Composition patterns: An approach to designing reusable aspects. In *Proceedings of the 23rd International Conference on Software Engineering*, ICSE '01, pages 5–14. IEEE Computer Society, 2001.

[11] Erich Gamma, Richard Helm, Ralph E. Johnson, and John Vlissides. *Design Patterns. Elements of Reusable Object-Oriented Software.* Addison-Wesley, March 1995.

[12] John Grundy. Multi-perspective specification, design and implementation of software components using aspects. *International Journal of Software Engineering and Knowledge Engineering*, 10(6):713–734, Dec 2000.

[13] Yan Han, Gnter Kniesel, and Armin B. Cremers. Towards visual aspectj by a meta model and modeling notation. In *Proceedings of 6th Aspect Oriented Modeling Workshop at AOSD*, 2005.

[14] J.L. Herrero, F. Sanchez, F. Lucio, and M. Torro. Introducing Separation of Aspects at Design Time. In *Aspect-Oriented Programming (AOP) Workshop at ECOOP 2000.* 2000.

[15] Ivar Jacobson and Pan-Wei Ng. *Aspect-Oriented Software Development with Use Cases*. Addison Wesley, 2004.
[16] Mohamed M. Kande, Jorg Kienzle, and Alfred Strohmeier. From AOP to UML- A Bottom-Up Approach. In *Proceedings of the 1st International Workshop on Aspect-Oriented Modeling with UML*. Enschede, The Netherlands, 2002.
[17] Mohamed M. Kande, Jorg Kienzle, and Alfred Strohmeier. From AOP to UML: Towards an Aspect-Oriented Architectural Modeling Approach. Technical report, Swiss Federal Institute of Technology, Lausanne, Switzerland, 2002.
[18] Gregor Kiczales, John Lamping, Anurag Menhdhekar, Chris Maeda, Cristina Lopes, Jean-Marc Loingtier, and John Irwin. Aspect-Oriented Programming. In *Proceedings European Conference on Object-Oriented Programming*, volume LNCS 1241, pages 220–242. Springer-Verlag, 1997.
[19] Francisca Losavio, Alfredo Matteo, and Patricia Morantes. UML Extensions for Aspect Oriented Software Development. *Journal of Object Technology*, 8(5):85–104, 2009.
[20] David L. Parnas. On the Criteria To Be Used in Decomposing Systems into Modules. *Communications of the ACM*, 15(12):1053–1058, December 1972.
[21] Renaud Pawlak, Laurence Duchien, Gerard Florin, Fabrice Legond-Aubry, Lionel Seinturier, and Laurent Martelli. A uml notation for aspect-oriented software design. In *Proceedings of the Aspect Oriented Modeling with UML workshop at AOSD (2002*, 2002.
[22] James Rumbaugh, Ivar Jacobson, and Grady Booch. *The Unified Modeling Language Reference Manual*. Addison-Wesley, 1999.
[23] Aspect Oriented Programming with Spring. http://docs.spring.io/spring/docs/4.0.3.RELEASE/spring-framework-reference/htmlsingle/#aop.
[24] Dominik Stein, Stefan Hanenberg, and Rainer Unland. An UML-based Aspect-Oriented Design Notation for AspectJ. In *AOSD 2002*, pages 1–7, 2002.
[25] Junichi Suzuki and Yoshikazu Yamamoto. Extending UML with Aspects: Aspect Support in the Design Phase. In *Aspect-Oriented Programming (AOP) Workshop at ECOOP'99*, pages 14–18. 1999.
[26] Peri L. Tarr, Harold Ossher, William H. Harrison, and Stanley M. Sutton Jr. N Degrees of Separation: Multi-Dimensional Separation of Concerns. In *Proceedings of the 21st International Conference on Software Engineering*, pages 107–119, May 1999.
[27] UML 2.4.1 Superstructure. http://www.omg.org/spec/UML/2.4.1/Superstructure/PDF/.
[28] Unified Modeling Language(UML). http://www.uml.org/.
[29] Jos Warmer and Anneke Kleppe. *The Object Constraint Language: Precise Modeling with UML*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1999.
[30] Aida Atef Zakaria, Hoda Hosny, and Amir Zeid. A uml extension for modeling aspect-oriented systems. In *Second International workshop on Aspect-Oriented Modeling with UML at UML 2002*. 2002.
[31] Gefei Zhang. Towards aspect-oriented class diagrams. In *Proceedings of 12th Asia-Pacific Software Engineering Conference*, pages 763–768, 2005.

Babeş-Bolyai University, Department of Computer Science, 1, M. Kogălniceanu street, 400084 Cluj-Napoca, Romania
E-mail address: grigo@cs.ubbcluj.ro
E-mail address: adriana@cs.ubbcluj.ro